AD-A074 800 ARMY RESEARCH OFFICE RESEARCH TRIANGLE PARK NC F/G 5/9

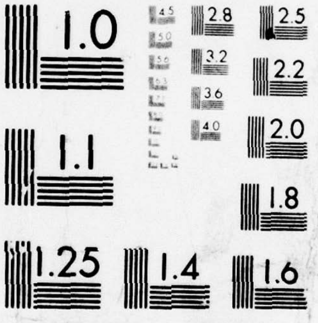
PROCEEDINGS OF THE 1979 ARMY NUMERICAL ANALYSIS AND COMPUTERS C--ETC(U)

SEP 79

UNCLASSIFIED ARO-79-3 NL

1 OF 6

AD
A074 800


1.0
4.5
5.0
5.6
6.3
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

14

ARO Report 79-3

6

PROCEEDINGS OF THE 1979 ARMY NUMERICAL ANALYSIS AND COMPUTERS CONFERENCE

held 14-16 February 1979,
White Sands Missile Range,
New Mexico.

9 Interim rept.

12 495

11 Sep 79

Approved for public release; distribution unlimited. The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

SPONSORED BY
THE ARMY MATHEMATICS STEERING COMMITTEE ON BEHALF OF

THE OFFICE OF
THE CHIEF OF RESEARCH, DEVELOPMENT AND
ACQUISITION

040 900

1530-1650 TECHNICAL SESSION II

Chairperson - Professor John Nohel, Mathematics Research Center, Madison, Wisconsin

1530-1550 REPRO-MODELING AND SIMULATION

Drs. Joseph T. Ryan and James L. Thompson, US Army Tank-Automotive Research and Development Command, Warren, Michigan

1550-1610 A NUMERICAL CHALLENGE: THE SOLUTION OF VARIATIONAL MODELS FOR MESOSCALE ATMOSPHERIC ANALYSIS

Dr. William D. Ohmstede, US Army Atmospheric Sciences Laboratory, White Sands Missile Range, New Mexico

1610-1630 A FORTRAN ROUTINE FOR ESTIMATING NORMAL DISTRIBUTION PARAMETERS

Dr. Bernard N. Goulet, US Army Materiel Systems Analysis Activity, Aberdeen Proving Ground, Maryland

1630-1650 A PROCEDURE FOR CONSOLIDATING LINE OF SIGHT DATA

Dr. Bernard N. Goulet, US Army Materiel Systems Analysis Activity, Aberdeen Proving Ground, Maryland

15 February 1979

Thursday Morning

0830-0930 GENERAL SESSION I

Chairperson - Dr. Stanley M. Taylor, Jr., Ballistic Research Laboratory, Aberdeen Proving Ground, Maryland

Speaker - Dr. Virginia Klema, Massachusetts Institute of Technology, Cambridge, Massachusetts

Title - ROBUST SOFTWARE FOR ROBUST STATISTICS

0930-1000 BREAK

1000-1200 TECHNICAL SESSION III

Chairperson - Mr. Morton Hirschberg, Ballistic Research Laboratory, Aberdeen Proving Ground, Maryland

14 February 1979

Wednesday Afternoon

1300-1345 REGISTRATION

1345-1400 WELCOMING AND OPENING REMARKS - Dr. Richard H. Duncan, Technical Director and Chief, White Sands Missile Range, White Sands Missile Range, New Mexico

1400-1500 KEYNOTE ADDRESS

Chairperson - Mr. Robert E. Green, Instrumentation Directorate, White Sands Missile Range, New Mexico

Speaker - Professor Gene H. Golub, Stanford University, Stanford, California

Title - NONLINEAR LEAST SQUARES AND SEPARATION OF VARIABLES

1500-1530 BREAK

1530-1650 TECHNICAL SESSION I

Chairperson - Dr. Theodore Hopp, Harry Diamond Laboratories, Adelphi, Maryland

1530-1550 DETERMINATION OF THE HIT AND KILL PROBABILITIES FOR SHOOTING THROUGH TREES

Professor Shelemyahu Zacks, USA TRASANA and Case Western Reserve University, Cleveland, Ohio

1550-1610 A COMPARISON OF VARIOUS METHODS FOR COMPUTING INSTANTANEOUS IMPACT PREDICTIONS OF MISSILES FOR RANGE FLIGHT SAFETY

Dr. Jerry F. Kuzanek, White Sands Missile Range, New Mexico

1610-1630 SURFACE MISS-DISTANCE PROGRAM

Mr. Dale McLaughlin, White Sands Missile Range, New Mexico

1630-1650 THE OTT FUNCTIONS FOR NAVAL GUNFIRE CONTROL SYSTEMS

Dr. Garland H. Ott, Naval Surface Weapons Center, Dahlgren, Virginia

TABLE OF CONTENTS*

*This Table of Contents lists only the papers that are published in this Technical Manual. For a list of all of the papers presented at the 1979 Army Numerical Analysis and Computers Conference see the copy of the Agenda.

Members of the AMSC would like to thank Mr. Green and his committee for the important role they played in making this an enjoyable and interesting scientific meeting. The physical location of the conference was in El Paso, Texas.

The theme of this Army-wide conference was "Data Analysis and Approximation". These areas are of special interest to several scientists at the host installation as well as many scientists throughout all the Army laboratories. The following list of invited speakers and the titles of their addresses shows that most of them related their topics to the theme.

Speakers and Institution	Area of Talk
Professor Gene H. Golub Stanford University	Nonlinear Least Squares and Separation of Variables
Dr. Virginia Klema Massachusetts Institute of Technology	Robust Software for Robust Statistics
Professor E. W. Cheney University of Texas-Austin	Approximating Functions of Two Variables by Functions of One Variable
Professor Philip W. Smith Texas A&M University	New Techniques in the Use of Splines for Data Analysis
Professors Richard A. Tapia with co-author James R. Thompson Rice University	Nonparametric Density Estimation: Where Do We Go From Here?

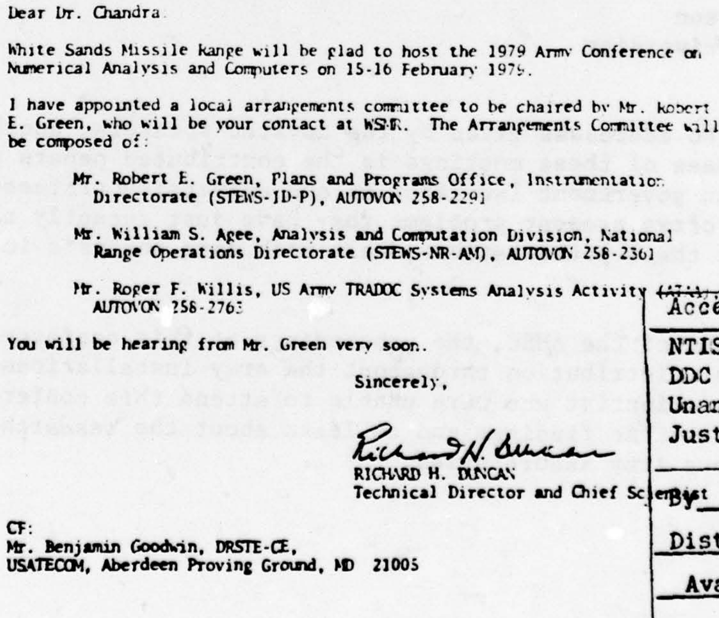
In addition to addresses given by the invited speakers, another very important phase of these meetings is the contributed papers presented by scientists in government laboratories and university professors. These individuals often present problems they have just recently solved. This year many of these presentations dealt with data analysis in one form or another.

At the request of the AMSC, the proceedings of this conference are being published for distribution throughout the Army installations. This will enable those scientist who were unable to attend this conference to benefit from the scientific findings and to learn about the research being conducted in the various Army laboratories.

FOREWORD

The application of set-theoretic language led, about 1900, to the discovery of various paradoxes. Therefore, it became necessary to limit the rules for the manipulation of arbitrary sets. It was suggested that one way of doing this was to set up explicit systems of axioms. But might not this procedure also lead to a new paradox? That is, how could one be certain a given set of axioms was consistent? In 1931, Kurt Godel proved that such consistency proofs are impossible for a system of mathematical logic. Several years later Godel pointed out that the steps in formal proofs could be replaced by operations with numbers. These step-by-step processes he called recursive. In 1936, Alan Turing identified such procedures, the general recursive functions, with outcomes of what could be computed by a machine. The problems of formal proofs became problems in the theory of computability! One sees from these remarks that the development of mathematical logic and proof theory in the early years of the twentieth century laid the foundations for the introduction of digital computers and the basis for their great transformation of our present day technology.

The Army Mathematics Steering Committee (AMSC) is the sponsor of the Numerical Analysis and Computers Conferences. Each year the conference is hosted by a different Army laboratory or installation. As chairman of the AMSC, Dr. Jagdish Chandra was therefore pleased to receive the following letter.



Dear Dr. Chandra

White Sands Missile Range will be glad to host the 1979 Army Conference on Numerical Analysis and Computers on 15-16 February 1979.

I have appointed a local arrangements committee to be chaired by Mr. Robert E. Green, who will be your contact at WSMR. The Arrangements Committee will be composed of:

Mr. Robert E. Green, Plans and Programs Office, Instrumentation Directorate (STEWS-ID-P), AUTOVON 258-2291

Mr. William S. Agee, Analysis and Computation Division, National Range Operations Directorate (STEWS-NR-AM), AUTOVON 258-2361

Mr. Roger F. Willis, US Army TRADOC Systems Analysis Activity, AUTOVON 258-2763

You will be hearing from Mr. Green very soon.

Sincerely,

Richard H. Duncan
RICHARD H. DUNCAN
Technical Director and Chief Scientist

CF:
Mr. Benjamin Goodwin, DRSTE-CE,
USATECOM, Aberdeen Proving Ground, MD 21005

U. S. ARMY RESEARCH OFFICE

Report No. 79-3

September 1979

PROCEEDINGS OF THE 1979 ARMY NUMERICAL ANALYSIS AND COMPUTERS CONFERENCE, held

Sponsored by the Army Mathematics Steering Committee

HOST

U. S. ARMY WHITE SANDS MISSILE RANGE

White Sands Missile Range, New Mexico

14-16 February 1979

Approved for public release; distribution unlimited. The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

U. S. Army Research Office
P. O. Box 12211
Research Triangle Park, North Carolina

1000-1020 ADJUSTMENT BY LEAST SQUARES IN GEODESY USING BLOCK ITERATIVE METHODS FOR SPARSE MATRICES

Professor Robert J. Plemmons, The University of Tennessee, Knoxville, Tennessee

1020-1040 A ROUTINE FOR NUMERICAL EVALUATION OF INTEGRALS WITH OSCILLATORY BEHAVIOR

Dr. Theodore Hopp, Harry Diamond Laboratories, Adelphi, Maryland

1040-1100 COMPUTER AIDED SOLUTION OF THE BACTERIAL SURVIVAL EQUATIONS IN MICROBIOLOGY

Drs. Chia Ping Wang and Ari Brynjolfsson, US Army Natick Research and Development Command, Natick, Massachusetts

1100-1120 NASTRAN ANALYSIS OF A PHOTOPLASTIC MODEL FOR SLIDING BREECH MECHANISM

Drs. P. C. T. Chen and G. P. O'Hara, US Army Armament Research and Development Command, Benet Weapons Laboratory, Watervliet, New York

1120-1140 A COMPUTER PROGRAM AND APPROXIMATE SOLUTION FORMULATION FOR GUN MOTIONS ANALYSIS

Dr. Julian J. Wu, US Army Armament Research and Development Command, Benet Weapons Laboratory, Watervliet, New York

1140-1200 PRELIMINARY INVESTIGATION OF THE SINGULAR BEHAVIOR OF FLUIDS NEAR A SLIDING CORNER

Dr. Patrick J. Roache, Ecodynamics Research Associates, Inc., Albuquerque, New Mexico and Dr. Csaba K. Zoltani, Ballistic Research Laboratory, Aberdeen Proving Ground, Maryland

1000-1200 <u>TECHNICAL SESSION IV</u>

Chairperson - Dr. James A. Graves, Plans Office, White Sands Missile Range, New Mexico

1000-1020 NUMERICAL SOLUTION OF THE SAD METHOD FOR ROBUST REGRESSION

Messrs. William S. Agee and Robert H. Turner, White Sands Missile Range, New Mexico

1020-1040 HOW DOES AGEE'S SMOOTHING METHOD WORK?

Professor Carl de Boor, Mathematics Research Center, Madison, Wisconsin

1040-1100 THE GIFT COMPUTER CODE

Dr. Gary G. Kuehl, Ballistic Research Laboratory, Aberdeen Proving Ground, Maryland

1100-1120 RESOLUTION OF AMBIGUITY FOR RESIDUE MEASUREMENT BY ANGLE MEASUREMENT EQUIPMENT

Mr. Lloyd Shepherd and Mr. John W. Starner, Jr., White Sands Missile Range, New Mexico

1120-1140 A COMPARISON OF THE EXISTENCE THEOREMS OF KANTOROVICH AND MOORE

Professor Louis B. Rall, Mathematics Research Center, Madison, Wisconsin

1140-1200 CURVE-FOLLOWING TECHNIQUES FOR SOLVING A SET OF NONLINEAR EQUATIONS

Professor T. Y. Li, Mathematics Research Center, Madison, Wisconsin

Thursday Afternoon

1200-1330 LUNCH

1330-1510 TECHNICAL SESSION V

Chairperson - Dr. P. C. T. Chen, Benet Weapons Laboratories,
Watervliet Arsenal, Watervliet, New York

1330-1350 ONE DIMENSIONAL SHOCK FLOW CALCULATIONS WITH ACCURACY x/20

Professors James Glimm and Dan Marchesin, The Rockefeller University, New York, New York

1350-1410 COLLOCATION AT GAUSS POINTS AS A DISCRETIZATION IN OPTIMAL CONTROL

Professor G. W. Reddien, Vanderbilt University, Nashville, Tennessee

1410-1430 ON THE STABILITY OF CERTAIN DISCRETIZATIONS ON NONUNIFORM MESHES

Professor Eusebius J. Doedel, Vanderbilt University, Nashville, Tennessee

1430-1450 FAST APPROXIMATIONS FOR RICCATI EQUATIONS VIA WALSH FUNCTIONS AND BLOCK PULSE FUNCTIONS

Mr. Oren N. Dalton, White Sands Missile Range, New Mexico

1450-1510 THE NUMERICALLY STABLE CONSTRUCTION OF PERIODIC JACOBI MATRICES WITH PRESCRIBED SPECTRAL PROPERTIES

Professor Warren Ferguson, Mathematics Research Center, Madison, Wisconsin

1510-1540 BREAK

1540-1640 GENERAL SESSION II

Chairperson - Dr. James L. Thompson, US Army Tank-Automotive Research and Development Command, Warren, Michigan

1540-1610 Speaker - Professor E. W. Cheney, University of Texas, Austin, Texas
Title - APPROXIMATING FUNCTIONS OF TWO VARIABLES BY FUNCTIONS OF ONE VARIABLE

1610-1640 Speaker - Professor Philip W. Smith, Texas A&M University, College Station Texas
Title - NEW TECHNIQUES IN THE USE OF SPLINES FOR DATA ANALYSIS

16 February 1979

Friday Morning

0830-0900 GENERAL SESSION III

Chairperson - Dr. Chia Ping Wang, US Army Natick Research and Development Command, Natick, Massachusetts

Speaker - Professors Richard A. Tapia and James R. Thompson (co-author), Rice University, Houston, Texas

Title - NONPARAMETRIC DENSITY ESTIMATION: WHERE DO WE GO FROM HERE?

0900-1030 OPEN MEETING OF THE NUMERICAL ANALYSIS AND COMPUTERS SUBCOMMITTEE

Chairperson - Paul T. Boggs, US Army Research Office, Research Triangle Park, North Carolina

1030 ADJOURN

Extensions and Uses of the Variable Projection Algorithm for Solving Nonlinear Least Squares Problems

Gene H. Golub[*]
Randall J. LeVeque[*]

Abstract. The variable projection algorithm for solving separable nonlinear least squares problems with a single data vector is well known[1]. We review that theory and present a modification of the algorithm for solving problems in which it is desired to fit more than one data vector with the same nonlinear parameters (though possibly different linear parameters) for each right hand side. We give an example from chemical kinetics and also show how such problems arise from an inverse differential equations problem as in compartmental analysis. A further modification is presented for dealing with total least squares problems: problems in which the independent variables as well as the observartions may have errors.

[*] Department of Computer Science, Stanford University, Stanford, CA 94305.

This work was supported in part by Army contract No. DAAG29-78-G-0179, Department of Energy contract No. EY-76-S-03-0326 PA#30 and by a National Science Foundation graduate fellowship. The paper was produced using TEX, a computer typesetting system created by Donald Knuth at Stanford.

1. Introduction.

The variable projection algorithm [1] has gained much popularity in recent years as a method for solving separable nonlinear least squares problems. A separable problem is one in which the model can be written in the form

$$\eta(t) \approx \sum_{j=1}^{n} \beta_j \phi_j(\alpha; t); \qquad \alpha \in \mathbb{R}^k, \tag{1.1}$$

where the ϕ_j are given functions of α and t and $\approx$ indicates approximation in the least squares sense. Given data $y = (\eta_1, \eta_2, \ldots, \eta_m)^T$ observed at times $t_1, t_2, \ldots, t_m$ respectively, the problem consists of finding the optimal parameters $\hat{b} = (\hat{\beta}_1, \ldots, \hat{\beta}_n)^T$, $\hat{\alpha} = (\hat{\alpha}_1, \ldots, \hat{\alpha}_k)^T$ which minimize the sum of the squares of the residuals, given by

$$\sum_{i=1}^{m} \left(\eta_i - \sum_{j=1}^{n} \beta_j \phi_j(\alpha; t_i) \right)^2 .$$

If we let the matrix $\Phi(\alpha)$ consist of the components $\phi_j(\alpha; t_i)$, $i = 1, 2, \ldots, m$; $j = 1, 2, \ldots, n$, then the problem can be restated as

$$\text{minimize} \quad \|y - \Phi(\alpha) b\|^2 \qquad \text{over } b \in \mathbb{R}^n,\ \alpha \in \mathbb{R}^k.$$

Here we have used the 2-norm, $\|x\|^2 = \sum \xi_i^2$. The theory presented here can be easily extended to the case of weighted norms, $\|x\|_W^2 = \sum w_i \xi_i^2$.

The separable least squares problem which is probably most often encountered in practice is that of exponential fitting. In this case we want to fit

$$\eta_i \approx \sum_{j=1}^{n} \beta_j e^{\alpha_j t_i}, \qquad i = 1, 2, \ldots, m.$$

This leads to a matrix $\Phi(\alpha)$ of the form

$$\Phi(\alpha) = \begin{pmatrix} e^{\alpha_1 t_1} & \ldots & e^{\alpha_n t_1} \\ \vdots & & \vdots \\ e^{\alpha_1 t_m} & \ldots & e^{\alpha_n t_m} \end{pmatrix}.$$

In the next section we review the variable projection algorithm as modified by Kaufman[2]. We then present an extension of the algorithm to handle multiple right hand sides and discuss some uses of the algorithm in solving inverse differential equations and total least squares problems.

2. The variable projection algorithm.

For any fixed a, the problem of finding the optimal b corresponding to that a is a linear least squares problem. A solution is given by

$$b = \Phi^{+}(a)y$$

where $\Phi^{+}(a)$ is the pseudo-inverse of $\Phi(a)$. The idea behind the variable projection algorithm is to define the functional $r(a)$ by

$$\begin{aligned} r(a) &= \min_{b} \|y - \Phi(a)b\|^2 \\ &= \|y - \Phi(a)\Phi^{+}(a)y\|^2. \end{aligned} \tag{2.1}$$

This functional is then minimized with respect to a by means of the Levenberg-Marquardt algorithm (described below) to give the optimal $\hat{a}$. The linear parameters $\hat{b}$ are then recovered from $\hat{b} = \Phi^{+}(\hat{a})y$. It can be shown[1] that this method of solution yields the correct result provided that $\Phi(a)$ has constant rank in a neighborhood of the solution $\hat{a}$. The advantage of this technique is that the nonlinear functional $r(a)$ which must be minimized is now a function only of a. The parameters β_i do not appear explicitly and so the size of our nonlinear problem has been reduced.

For any given a, there is an orthogonal matrix

$$Q(a) = \begin{pmatrix} Q_1(a) \\ Q_2(a) \end{pmatrix}$$

with

$$\begin{aligned} &Q^T(a)Q(a) = I, \\ &Q_1(a) \in \mathbb{R}^{n\times m}, \\ &Q_2(a) \in \mathbb{R}^{(m-n)\times m}, \end{aligned}$$

such that $Q(a)$ triangularizes $\Phi(a)$, that is,

$$\begin{pmatrix} Q_1(a) \\ Q_2(a) \end{pmatrix} \Phi(a) = \begin{pmatrix} U_1(a) \\ 0 \end{pmatrix}, \tag{2.2}$$

with $U_1(a)$ right triangular. From the invariance of the 2-norm under orthogonal transformations, we have that

$$\begin{aligned} \|y - \Phi(a)b\|^2 &= \|Q(a)(y - \Phi(a)b)\|^2 \\ &= \left\| \begin{pmatrix} Q_1(a)y \\ Q_2(a)y \end{pmatrix} - \begin{pmatrix} U_1(a) \\ 0 \end{pmatrix} b \right\|^2 \\ &= \|Q_1(a)y - U_1(a)b\|^2 + \|Q_2(a)y\|^2. \end{aligned}$$

Since the optimal b for any a is $b = \Phi^{+}(a)y = U_1^{-1}(a)Q_1(a)y$, the residual $r(a)$ is simply

$$r(a) = \|Q_2(a)y\|^2.$$

It is this form of r to which we apply the minimization algorithm.

The Levenberg-Marquardt algorithm is a general iterative procedure for minimizing $\|f(a)\|^2$ for a nonlinear m-vector valued function f. At the j^{th} stage we have an approximation $a^{(j)}$ to the solution and we compute the Jacobian $J(a^{(j)})$ of f,

$$J(a^{(j)}) = \left(\frac{\partial f(a^{(j)})}{\partial a_1}, \ldots, \frac{\partial f(a^{(j)})}{\partial a_k}\right),$$

where

$$\frac{\partial f}{\partial a_j} = \left(\frac{\partial f_1}{\partial a_j}, \ldots, \frac{\partial f_m}{\partial a_j}\right)^T.$$

We also choose a value for the "Marquardt parameter" ν_j and then solve the system

$$\begin{pmatrix} J(a^{(j)}) \\ \nu_j I \end{pmatrix} \delta^{(j)} \approx \begin{pmatrix} f(a^{(j)}) \\ 0 \end{pmatrix}$$

in the least squares sense for the correction $\delta^{(j)}$. The next iterate $a^{(j+1)}$ is then obtained as $a^{(j+1)} = a^{(j)} - \delta^{(j)}$. The parameter ν_j is used to control the length of the correction vector $\delta^{(j)}$. The correction $\delta^{(j)}$ minimizes

$$\|J(a^{(j)})\delta^{(j)} - f\|^2 + \|\nu_j \delta^{(j)}\|^2.$$

If $\nu_j = 0$, this becomes simply the Gauss-Newton algorithm. For further discussion of this minimization technique, see [5].

In the present context, $f(a) = Q_2(a)y$, so in order to apply this algorithm we must be able to compute the columns of the Jacobian matrix,

$$\frac{\partial(Q_2(a)y)}{\partial a_j} = \frac{\partial Q_2(a)}{\partial a_j} y.$$

Recall from (2.2) that $Q_2(a)\Phi(a) = 0$, so by differentiating we get

$$\frac{\partial Q_2(a)}{\partial a_j}\Phi(a) = -Q_2(a)\frac{\partial \Phi(a)}{\partial a_j}.$$

Golub and Pereyra[1] give an exact expression for $\partial Q_2(a)/\partial a_j$, but Kaufman[2] has suggested the simplification

$$\frac{\partial Q_2(a)}{\partial a_j} \approx -Q_2(a)\frac{\partial \Phi(a)}{\partial a_j}\Phi^+(a) \qquad (2.3)$$

which works well in practice.

3. Extensions to multiple right hand sides.

Occasionally one wishes to solve a problem in which there are a number of vectors $y_1, y_2, \ldots, y_s$ of data each of which is to be fit by a model of the form (1.1). We allow the linear parameters b to be different for each data vector. If the nonlinear parameters a are also allowed to be different for each y_i, then there is no problem. We are simply faced with s distinct problems of the type already discussed. If however, a is constrained to be the same for each data vector, then we have a new problem. In this case, we wish to fit the matrix $Y = (y_1, y_2, \ldots, y_s) \in \mathbb{R}^{m \times s}$ by a matrix of the form $\Phi(a)B$ where $\Phi(a)$ is as before and $B \in \mathbb{R}^{n \times s}$. The minimization problem is now

$$\text{minimize} \quad \|Y - \Phi(a)B\|_F^2 \qquad \text{over } a \in \mathbb{R}^k,\ B \in \mathbb{R}^{n \times s}. \qquad (3.1)$$

Here we use the Frobenius norm of the matrix, $\|X\|_F^2 = \sum_{i,j} \xi_{ij}^2$.

One example of this type of problem occurs in the analysis of data from the biological substance bacteriorhodopsin. The data vectors y_i consist of measurements of the amount of light absorbed by the substance at m different times during the course of a chemical reaction. We have s different data vectors, one for each of several wavelengths of light used. In this case the kinetic theory dictates that for each wavelength the absorbtion curve should be a sum of exponentials, and that the rate constants a should be the same for all wavelengths. Hence the data Y must be fit by a model of the form $\Phi(a)B$ where $\phi_j(a; t_i) = e^{a_j t_i}$.

One approach to solving this problem would be to write

$$\bar{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_s \end{pmatrix} \in \mathbb{R}^{ms}, \qquad \bar{b} = \begin{pmatrix} b_1 \\ \vdots \\ b_s \end{pmatrix} \in \mathbb{R}^{ns},$$

$$\bar{\Phi}(a) = \begin{pmatrix} \Phi(a) & & & \\ & \Phi(a) & & \\ & & \cdots & \\ & & & \Phi(a) \end{pmatrix} \in \mathbb{R}^{ms \times ns} \qquad (3.2)$$

and then minimize $\|\bar{y} - \bar{\Phi}\bar{b}\|^2$ by means of the algorithm as previously discussed. However, the matrix $\bar{\Phi}(a)$ is excessively large. The problem (3.1) has a special structure which is not sufficiently exploited by this approach.

Instead, for any a we proceed as before by finding $Q(a)$ such that (2.2) holds. Since the Frobenius norm, like the 2-norm, is invariant under orthogonal transformations, we are led as before to the problem of simply minimizing $\|Q_2(a)Y\|_F^2$. But $\|Q_2(a)Y\|_F^2 = \|z(a)\|^2$ where the vector $z(a)$ is defined by

$$z(a) = \begin{pmatrix} Q_2(a)y_1 \\ \vdots \\ Q_2(a)y_s \end{pmatrix}. \tag{3.3}$$

The Levenberg-Marquardt algorithm previously described can be applied directly to this problem, provided we can compute $\partial z(a)/\partial a_j$ for $j = 1, 2, \ldots, k$. But from (3.3) it is clear that

$$\frac{\partial z(a)}{\partial a_j} = \begin{pmatrix} \frac{\partial Q_2(a)}{\partial a_j} y_1 \\ \vdots \\ \frac{\partial Q_2(a)}{\partial a_j} y_s \end{pmatrix},$$

where $\partial Q_2(a)/\partial a_j$is computed exactly as before.

So, in summary, at the j^{th} step of the minimization procedure we solve for the correction term $\delta^{(j)}$ from the linear system

$$\begin{pmatrix} \frac{\partial Q_2(a^{(j)})}{\partial a_1} y_1 & \cdots & \frac{\partial Q_2(a^{(j)})}{\partial a_k} y_1 \\ \vdots & & \vdots \\ \frac{\partial Q_2(a^{(j)})}{\partial a_1} y_s & \cdots & \frac{\partial Q_2(a^{(j)})}{\partial a_k} y_s \\ \hline & \nu_j I & \end{pmatrix} \delta^{(j)} = \begin{pmatrix} Q_2(a^{(j)})y_1 \\ \vdots \\ Q_2(a^{(j)})y_s \\ \hline 0 \end{pmatrix}.$$

Note that this system is the same size as that which would result from using the alternative formulation (3.2). The advantage of the latter approach lies in the calculation of $Q_2(a)$ and its derivatives. Here $Q_2(a) \in \mathbb{R}^{(m-n)\times m}$ whereas (3.2) would lead to an $(m-n)s$ by ms matrix.

This algorithm has been applied to the bacteriorhodopsin problem described earlier with satisfactory results. For that problem, we had $m \approx 175$, $s = 5$. The correct number of exponential terms was not known *a priori*, so fits were computed with 3, 4, 5 and 6 terms. Figure 1 shows the residuals for one such calculation. For more information on this particular application, see [4].

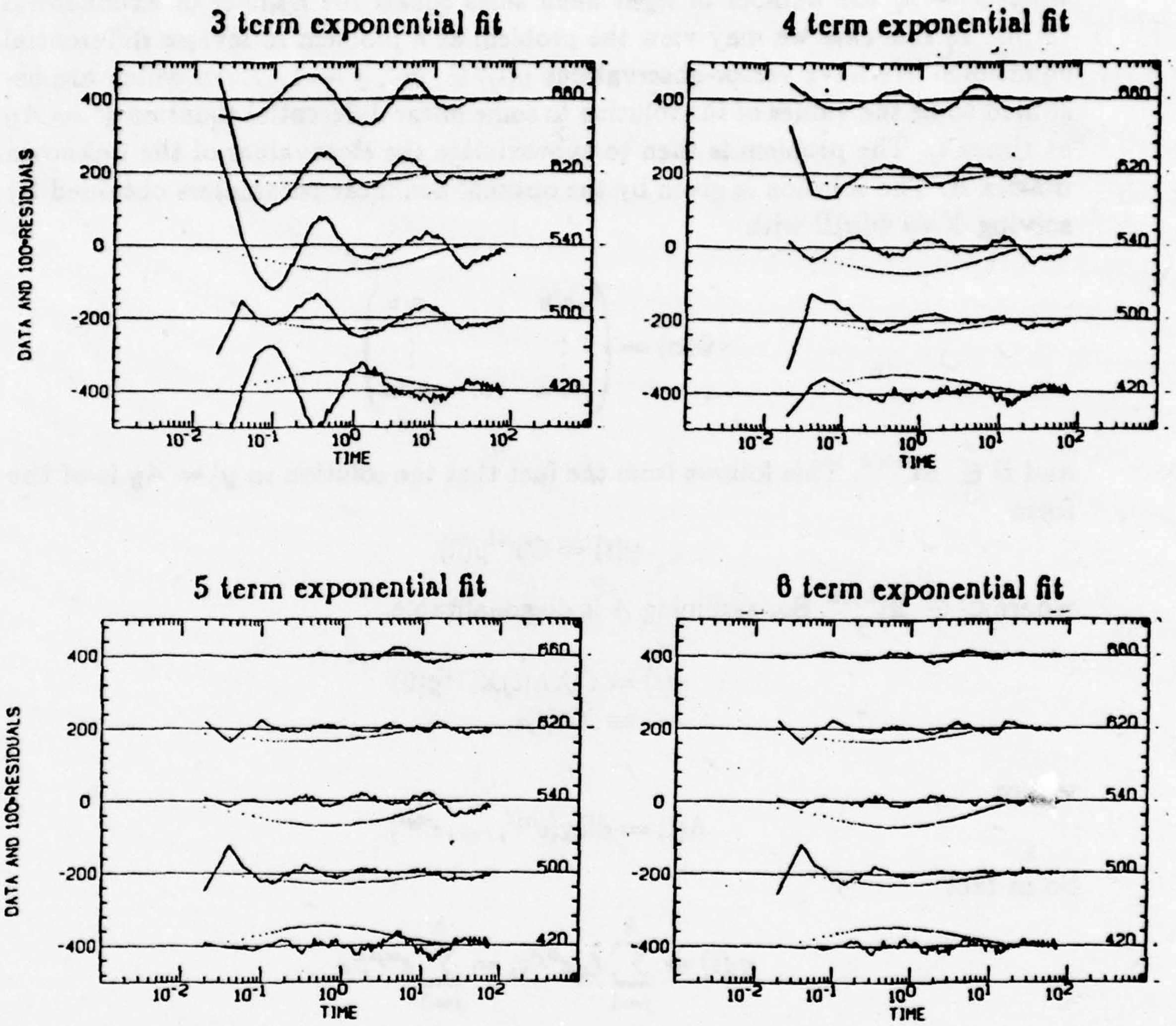


Figure 1
Bacteriorhodopsin data and magnified residuals

The data at each wavelength $\lambda = 420$ through $\lambda = 660$ is shown as points. The solid line is the residual from the fit obtained, multiplied by a factor of 100 to make it visible.

4. Inverse differential equations.

A special case of the exponential fitting problem of the previous section occurs when $s = k$, the number of right hand sides equals the number of exponential terms. In this case we may view the problem as a problem in inverse differential equations. We have vector-observations $y(t_j) \in \mathbb{R}^k$, $j = 1, \ldots, m$ which are assumed to be the values of the solution to some linear differential equation $y' = Ay$ at times t_j. The problem is then to approximate the eigenvalues of the unknown matrix A. The solution is given by the optimal nonlinear parameters obtained by solving $Y = \Phi(a)B$ with

$$\Phi(a) = \begin{pmatrix} e^{a_1 t_1} & \cdots & e^{a_k t_1} \\ \vdots & & \vdots \\ e^{a_1 t_m} & \cdots & e^{a_k t_m} \end{pmatrix}$$

and $B \in \mathbb{R}^{k \times k}$. This follows from the fact that the solution to $y' = Ay$ is of the form

$$y(t) = Ce^{At}y(0)$$

where $C \in \mathbb{R}^{k \times k}$. So, assuming A is diagonalizable,

$$\begin{aligned} y(t) &= CX\Lambda(t)X^{-1}y(0) \\ &\equiv \tilde{X}\Lambda(t)z, \end{aligned}$$

where

$$\Lambda(t) = \operatorname{diag}(e^{a_1 t}, \ldots, e^{a_k t}).$$

So in fact

$$\eta_i(t) = \sum_{j=1}^{k} \tilde{\xi}_{ij} e^{a_j t} z_j = \sum_{j=1}^{k} e^{a_j t} \beta_{ji}$$

where

$$\beta_{ji} = \tilde{\xi}_{ij} z_j.$$

5. Total least squares.

Up to this point we have assumed that the values of the independent variable t were known exactly. In that case our task was to minimize $\|Y - \Phi(a;t)B\|_F^2$. Note that we have written $\Phi(a;t)$ to show the dependence of Φ on t. For now we will consider the problem in which t as well as Y may have errors. In this case we wish to solve a problem of the form

$$\begin{aligned} &\text{minimize} \quad \left(\|Y - \Phi(\alpha;\tau)B\|_F^2 + \lambda\|\tau - t\|^2\right) \\ &\text{over } \alpha \in \mathbb{R}^k,\ B \in \mathbb{R}^{k\times s},\ \tau \in \mathbb{R}^m. \end{aligned} \qquad (5.1)$$

We can consider $a = \binom{\alpha}{\tau}$ as the vector of unknown nonlinear parameters. In order for the estimate of a to be consistent, the parameter λ must be chosen in proportion to the ratio of variances of the errors. For more generality, the norms in (6.1) can be weighted norms without introducing any difficulties.

Again there are two possible approaches to solving this problem. The first approach uses the variable projection algorithm as presented in section 2 on the problem

$$\text{minimize} \quad \|\bar{\Phi}(\alpha;\tau)\bar{B} - \tilde{Y}\|_F^2 \qquad \text{over } a, \bar{B}$$

where

$$\bar{\Phi}(\alpha;\tau) = \begin{pmatrix} \Phi(\alpha;\tau) & 0 \\ 0 & \tau \end{pmatrix} \begin{matrix} \}m \\ \}m \end{matrix}$$

$$\bar{B} = \begin{pmatrix} B \\ \hline 1 \;\cdots\; 1 \end{pmatrix}, \quad \tilde{Y} = \begin{pmatrix} Y \\ \hline t \;\cdots\; t \end{pmatrix}.$$

This is straightforward but requires that we be able to fix some of the linear parameters at 1. The variable projection algorithm can be easily modified to impose this constraint. Such a modification was first proposed by Krogh[3].

The second method which can be used for solving the total least squares problem involves a further modification of the functional to be minimized. Let $Q(a) = \begin{pmatrix} Q_1(a) \\ Q_2(a) \end{pmatrix}$ be the orthogonal matrix which triangularizes $\Phi(\alpha;\tau)$. Then we wish to find a to minimize

$$r(a) = \|Q_2(a)Y\|_F^2 + \lambda\|\tau - t\|^2.$$

This residual functional can be rewritten in partitioned from as

$$r(a) = \|[Q_2(a)Y \mid \lambda(\tau - t)]\|_F^2.$$

The Levenberg-Marquardt algorithm can be used for this minimization. Recall from section 2 that we require the derivatives of the columns of $[Q_2(a)Y \mid \lambda(\tau - t)]$

with respect to each nonlinear parameter. The derivatives of $Q_2(a)y_j$ with respect to any a_i or τ_i can be computed exactly as before. For the final column we have

$$\frac{\partial(\tau - t)}{\partial a_i} = 0,$$

$$\frac{\partial(\tau - t)}{\partial \tau_i} = e_i, \quad \text{the } i^{\text{th}} \text{ unit vector.}$$

So we see that at each stage of the minimization process we must solve for the corrections

$$\delta^{(j)} = \begin{pmatrix} \delta_a^{(j)} \\ \delta_\tau^{(j)} \end{pmatrix}$$

from the system

$$\left(\begin{array}{ccc|ccc} \frac{\partial Q_2(a^{(j)})}{\partial a_1^{(j)}} y_1 & \cdots & \frac{\partial Q_2(a^{(j)})}{\partial a_k^{(j)}} y_1 & \frac{\partial Q_2(a^{(j)})}{\partial \tau_1^{(j)}} y_1 & \cdots & \frac{\partial Q_2(a^{(j)})}{\partial \tau_m^{(j)}} y_1 \\ \vdots & & \vdots & \vdots & & \vdots \\ \frac{\partial Q_2(a^{(j)})}{\partial a_1^{(j)}} y_s & \cdots & \frac{\partial Q_2(a^{(j)})}{\partial a_k^{(j)}} y_s & \frac{\partial Q_2(a^{(j)})}{\partial \tau_1^{(j)}} y_s & \cdots & \frac{\partial Q_2(a^{(j)})}{\partial \tau_m^{(j)}} y_s \\ \hline \multicolumn{3}{c|}{0} & \multicolumn{3}{c}{I} \\ \hline \multicolumn{6}{c}{\nu_j I} \end{array}\right) \begin{pmatrix} \delta_a^{(j)} \\ \delta_\tau^{(j)} \end{pmatrix} = \left(\begin{array}{c} Q_2(a^{(j)}) y_1 \\ \vdots \\ Q_2(a^{(j)}) y_s \\ \hline \tau^{(j)} - t \\ \hline 0 \end{array}\right).$$

Both of the approaches described here were used on an example from Powell and Macdonald[7]. The set of data was originally given by Pearson[6] and the weights by York[9]. This data is given in Table 1. In this example both the times t_i and the observations y_i are weighted, by v_i and w_i respectively. This data was fitted by a linear polynomial $y \approx \beta_1 + \beta_2 x$, as was done in [7]. The results, $\beta_1 = 5.4799$, $\beta_2 = -0.48053$, agree with those reported there. The t_i were used as initial approximations to the nonlinear parameters and convergence to the accuracy shown occured in 5 iterations. The resulting fit is shown in figure 2. The given data points (t_i, y_i) are shown by X's, the points (τ_i, y_i) by boxes. Note the effect of the weights.

Table 1

Pearson's data and York's weights				
i	t_i	v_i	y_i	w_i
1	0.0	1000.0	5.9	1.0
2	0.9	1000.0	5.4	1.8
3	1.8	500.0	4.4	4.0
4	2.6	800.0	4.6	8.0
5	3.3	200.0	3.5	20.0
6	4.4	80.0	3.7	20.0
7	5.2	60.0	2.8	70.0
8	6.1	20.0	2.8	70.0
9	6.5	1.8	2.4	100.0
10	7.4	1.0	1.5	500.0

Total Least Squares fit to Pearson's Data with York's Weights

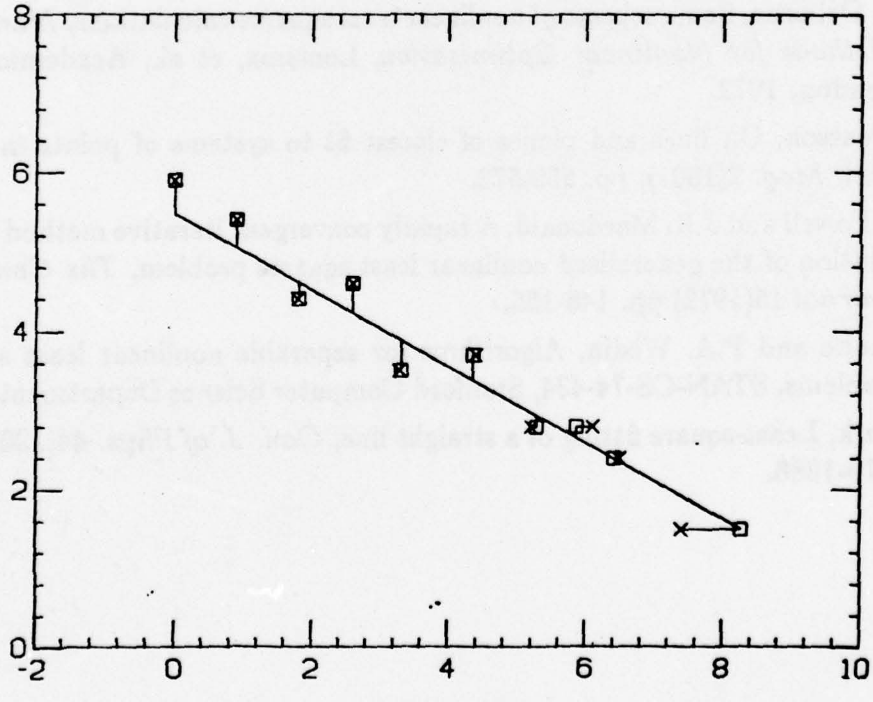


Figure 2

6. Acknowledgements.

The authors are indebted to Richard Lozier of the University of California at San Francisco for his support and expertise in the bacteriorhodopsin problem. Computer time for this research was provided by the Stanford Linear Accelerator Center of the U.S. Department of Energy.

7. References.

[1] G.H. Golub and V. Pereyra, The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate, SINUM 10(1973), pp.413-432.

[2] L. Kaufman, A variable projection method for solving separable nonlinear least squares problems, BIT 15(1975), pp. 49-57.

[3] F.T. Krogh, Efficient implementation of a variable projection algorithm for nonlinear least squares problems, CACM 17(1974), pp. 167-169.

[4] R. Lozier, R. LeVeque, G. Golub, Simultaneous analysis of kinetic data obtained at several monitoring wavelengths: applications to the bacteriorhodopsin photocycle, to appear.

[5] M.R. Osborne, Some aspects of nonlinear least squares calculations, *Numerical Methods for Nonlinear Optimization*, Lootsma, et al., Academic Press, London, 1972

[6] K. Pearson, On lines and planes of closest fit to systems of points in space, *Phil. Mag.* 2(1901), pp. 559-572.

[7] D.R. Powell and J.R. Macdonald, A rapidly convergent iterative method for the solution of the generalised nonlinear least squares problem, *The Computer Journal* 15(1972) pp. 148-155.

[8] A. Ruhe and P.A. Wedin, Algorithms for separable nonlinear least squares problems, STAN-CS-74-434, Stanford Computer Science Department, 1974.

[9] D. York, Least-square fitting of a straight line, *Can. J. of Phys.* 44(1966), pp. 1079-1086.

DETERMINATION OF THE HIT AND KILL PROBABILITIES FOR SHOOTING THROUGH TREES[+]

S. Zacks
Department of Mathematics and Statistics
Case Western Reserve University
Cleveland, Ohio 44106

ABSTRACT. The present paper provides a methodology for the computation of the hit/kill probability of a high caliber weapon shooting in a forest. We assume that trees are distributed at random according to a Poisson Law. Bullets which hit trees with sufficient by high velocity can penetrate the trunks and continue towards the target. A model is provided for the determination of the distribution of the exit velocity of a bullet. Recursive method is given for the computation of successive exit distributions as functions of the initial (muzzle) velocity, the distances between the trees and their characteristics. On the basis of this recursive method the kill probabilities are computed. Numerical examples are provided as well as FORTRAN programs.

+) Research conducted under Contract DAAG29-76-D-0100 with Battelle Columbus, Subcontract D.O. No. 0856, for USA TRASANA.

1. INTRODUCTION

The present technical report provides a method of evaluating the degradation effects on large caliber weapons located in a forest and shooting at a target in the forest. The degradation effect is actuallv measured by the decrease in the hit and kill probabilities of these weapons, compared to those when there are no obstacles in the trajectories of the bullets. The obstacles considered here are randomly located trees of varying trunk size. The initial (muzzle) velocity of the bullets is sufficiently high to allow penetration through trees. However, the penetrating bullet loses energy and its exit velocity may be considerably smaller than the velocity at which it hits the tree. The hitting velocity depends on the initial velocity and on the distance of the tree from the origin. The exit velocity is, however, a random variable which depends on the length of the bullet's path through the trunk and the resistance of the wood (type of tree). The penetrating bullet may also be deflected from its original aimed path. In Section 2 we specify the physical model under consideration and the probabilistic assumptions. The distribution of the exit velocity is derived from this model analytically in Section 3. In Section 4 we develop a recursive method for the numerical determination of the consecutive distributions of the exit velocities from n trees ($n \geq 1$), which are located at specified distances $d_1, d_2, \ldots, d_n$ from each other, on the path of the bullet. This recursive method is particularly convenient for numerical analysis. In Section 5 we derive an analytic expression for the hit/kill probability of a round. This analytic expression requires, however, numerical methods of evaluation. We provide numerical methods which combine exact computations with some Monte Carlo estimation. This Monte Carlo estimation appears only in one stage of the numerical evaluation, replacing a complicated numerical integration. The method is not, however, a pure simulation procedure. It has the property that with a very small number of (independent) runs we attain estimates with very high precision. Two alternative procedures are compared with respect to their precision and required computing time. FORTRAN programs are provided in the appendices.

2. THE PHYSICAL AND THE PROBABILISTIC MODEL

Consider a weapon located at the origin, 0, and shooting at a target which is at range R [m]. The initial (muzzle) velocity of the bullet is v_0 [m/sec]. If there are no obstacles along the trajectory of the bullet, it will hit the target with probability $P_H(v_0,R)$. Given that the bullet hits the target, the kill probability depends on the velocity of hitting the target, which is $v_0\lambda(v_0,R)$ and on other possible factors. Let $P_K(v_0,R)$ designate the combined kill probability. This function is specified in each particular case according to the specific weapon and fighting conditions. Similarly the function $\lambda(v_0,R)$ depends on the type of weapon, etc. We will consider here, for the sake of simplicity, a linear decreasing function of R, $v_0\ \lambda(v_0,R) = v_0-\beta R$. This is a good approximation when the initial velocity, v_0, is high and R is not too large a fraction of the weapons maximum range. The method developed in the present paper can be easily generalized to other types of ballistic functions.

The problem of shooting in the forest is that of randomly placed obstacles (trees) along the path (trajectory) of the bullet. We assume that the trees are randomly located according to a Poisson Law, with a given density, μ[No. of trees/m^2]. Thus, if we consider a strip around the straight line connecting the origin with the target, of length R[m] and width 1[m], the number of trees to be found on this strip is a random variable, N, having a Poisson distribution with mean μR.

In case of $N = n$, $n \geq 1$, let $D_1, D_2, \ldots, D_n$ be the distances (in [m]) from the origin to the location of the center of the first tree; from the first tree to the center of the second, etc.

$$\sum_{i=1}^{n} D_i \leq R$$

It is well known that the location points (Figure 1) $\xi_1 = D_1$, $\xi_2 = D_1 + D_2$, ..., $\xi_n = D_1 + \ldots + D_n$, are random variables having a joint distribution like the order statistics in a sample of n independent and identically distributed random variables from a uniform distribution on [0,R] (H. A. David [1] pp. 80).

The trees are generally of varying size. We are actually concerned with the size of the trunk at a certain height, h, above the ground. For the purpose of modeling we assume that a cut along a horizontal plan yields a circle of radius T (Figure 2). This radius is generally a random variable with a specific distribution, $F_T(t)$. The methods developed in the present paper are for a fixed radius T. We discuss at the end how the numerical procedures can be extended to cover the case of varying radius. Notice that the length of the bullets path in the trunk is

$$L = 2\,(T^2 - u^2)^{1/2} \tag{2.1}$$

where u (the distance of the path from the center) is a random variable, having a uniform distribution on the interval (0,T). Accordingly, L is a random variable having, for a given T, a distribution function (c.d.f.)

$$P_L\,(x;T) = 1 - \left[1 - \frac{x^2}{4T^2}\right]^{1/2}, \quad 0 \leq x \leq 2T. \tag{2.2}$$

3. THE DISTRIBUTION OF THE EXIT VELOCITY

Given an initial velocity v_0 and a tree of radius T located at a distance D from the origin, the question is what is the distribution of the exit velocity, v_1, of a bullet going through that tree. We say that the exit velocity is zero if the bullet is absorbed in the tree.

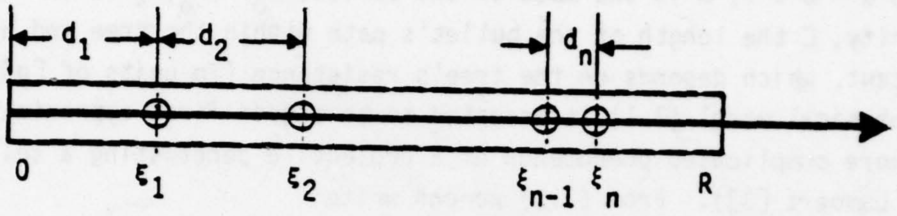


Figure 1. Random Location of Trees Along the Path of a Bullet

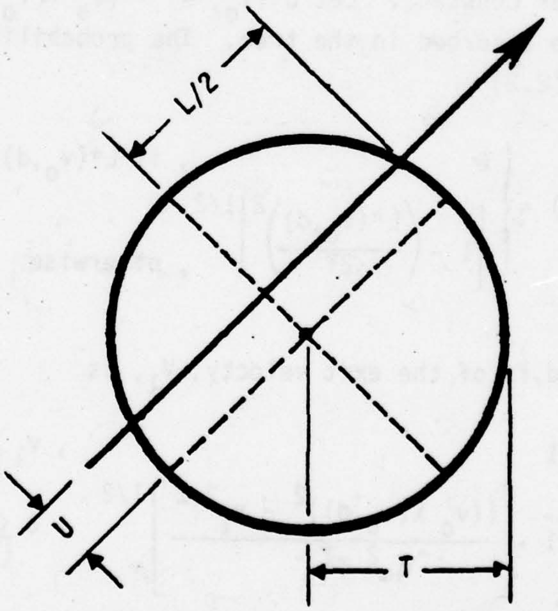


Figure 2. A Horizontal Cut of a Trunk of Radius T,
With the Bullet Path

The basic physical equation is that of energy conservation, namely

$$\frac{m}{2}\,(\lambda(v_0,d)v_0)^2 = \frac{m}{2}\,v_1{}^2 + \gamma L, \qquad (3.1)$$

where $d = D - T$, m is the mass of the bullet, $v_0\ \lambda(v_0,d)$ is the entrance velocity, L the length of the bullet's path within the tree and γ a proper constant, which depends on the tree's resistance (in units of $[g][m]/[sec]^2$). The physical model (3.1) is accepted to be a good first approximation to the more complicated phenomenon of a projectile penetrating a solid mass (see Lambert [3]). From (3.1) we can write

$$v_1{}^2 = (v_0\ \lambda(v_0,d))^2 - \alpha L, \qquad (3.2)$$

where α is a proper constant. Let $L^*(v_0,\ d) = (v_0\ \lambda(v_0,\ d))^2/\alpha$. If $L \geq L^*$ the bullet will be absorbed in the tree. The probability of this event is, according to (2.2)

$$q(v_0,d;\ T) = \begin{cases} 0 & ,\ \text{if } L^*(v_0,d) \geq 2T \\ \left[1 - \left(\dfrac{L^*(v_0,d)}{2T}\right)^2\right]^{1/2} & ,\ \text{otherwise} \end{cases} \qquad (3.3)$$

Similarly, the c.d.f. of the exit velocty, V_1, is

$$H(v_1;\ v_0,\ d_1,\ T) = \begin{cases} 1 & ,\ v_1 \geq v_0\ \lambda(v_0,\ d) \\ 1 - \left[\dfrac{((v_0\ \lambda(v_0,d))^2 - v_1{}^2)^2}{4\alpha^2\ T^2}\right]_+^{1/2} & ,\ o \leq v_1 \leq v_0\ \lambda(v_0,d) \end{cases} \qquad (3.4)$$

where $[a]_+ = \max(a,o)$. Notice that if T is small, $v_0^2\lambda^2(v_0,d) - v_1^2$ may be greater than $2\alpha T$.

In these cases the c.d.f. value is zero. Thus, let

$$v_1^*(v_0,d;T) = \left[(v_0\lambda(v_0,d))^2 - 2\alpha T\right]_+^{1/2}.$$

If $q(v_0,d;T) = 0$ then $v_1^*(v_0,d;T) \geq 0$. On the other hand, if $q(v_0,d;T) > 0$ then $v_1^*(v_0,d;T) = 0$. These relationships are illustrated in Figure 3.

4. RECURSIVE DETERMINATION OF THE SUCCESSIVE EXIT VELOCITY DISTRIBUTIONS

In the following we will adopt the simple model $v_0\lambda(v_0,d) = v_0-\beta d$. This assumption is not restrictive. The following formulae are developed for this particular function, since the data showed linearity in the region of interest. The formulae can be easily modified for other types of ballistic functions. Define, $H_1(x;v_0,d,T) = H(x;v_0,d,T)$, $0 \leq x \leq v-\beta d$.

4.1 The Case of n=2

Let d_1 and d_2 be the given distances. The distribution of the exit velocity from the second tree, V_2, can be obtained from $H_1(x;v_0,d,T)$, since the exit velocity from the first tree, if given, can be applied to compute the entrance velocity into the second tree. Accordingly, the c.d.f. of V_2, given v_0, d_1, d_2 and T is

$$H_2(v;\, v_0,d_1,d_2,T) = \int_{0-}^{v_0-\beta d_1} H_1(v;x,d_2,T)dH_1(x;v_0d_1,T) \qquad (4.1)$$

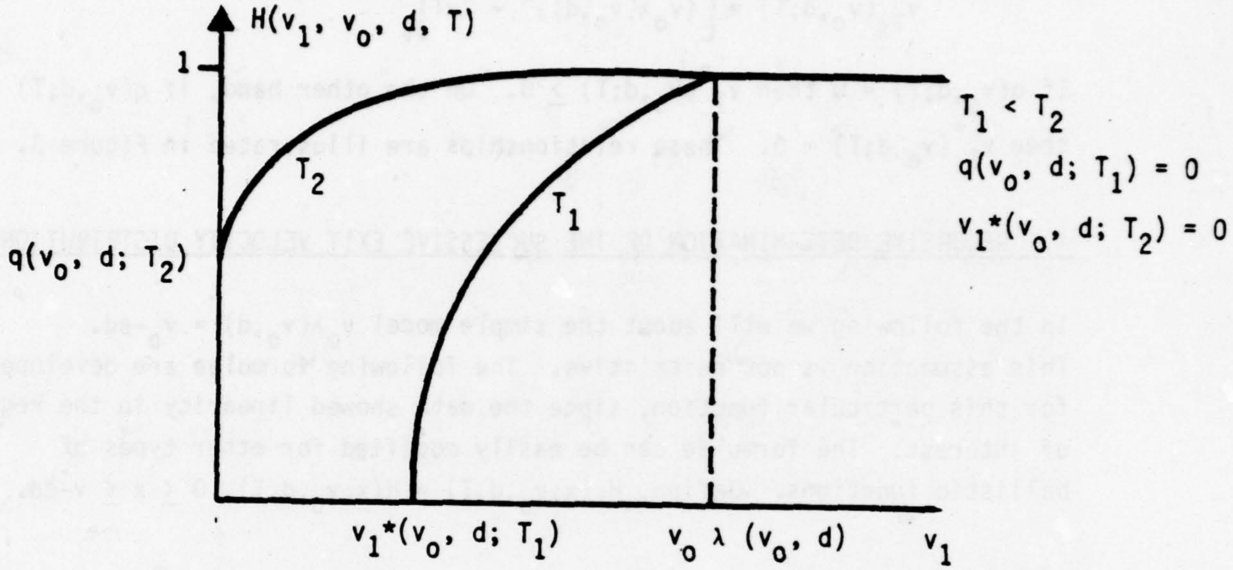


Figure 3. The C.D.F. of the Exit Velocity

Thus, we obtain after some manipulations

$$H_2(v;v_0,d_1,d_2,T) = \left[1 - \frac{[(v_0-\beta d_1-\beta d_2)^2 - v^2]^2}{4\alpha^2T^2}\right]_+^{1/2} \tag{4.2}$$

$$+ \frac{1}{2\alpha^2T^2}\int_v^{v_0-\beta d_1-\beta d_2} \left[1 - \frac{[(v_0-\beta d_1)^2 - (y+\beta d_2)^2]^2}{4\alpha^2T^2}\right]_+^{1/2} d_y \left[1 - \frac{(y^2 - v^2)^2}{4\alpha^2T^2}\right]_+^{1/2}$$

This integral should be understood as a regular integral over the range of values over which the functions within the squared brackets, [ ], are both positive. Furthermore,

$$d_y[G(y)]_+ = \begin{cases} 0 & \text{, if } G(y) \leq 0 \\ G'(y)d_y & \text{, if } G(y) > 0, \end{cases}$$

where $G'(y)$ is the derivative of $G(y)$. Our approach is to evaluate (4.2) numerically. We therefore leave it in its present form, without further analytical elaboration. For the purpose of approximating $H_2(v;v_0,d_1,d_2,T)$ numerically we partition the interval $(v, v_0-\beta d_1-\beta d_2)$ into M subintervals of equal size $\Delta = (v_0-\beta d_1-\beta d_2-v)/M$.

Define

$$\begin{aligned} \eta_i &= v + i\Delta \quad , i = 0, 1, \ldots M \\ \tilde{\eta}_i &= (\eta_i + \eta_{i-1})/2 \ , \ i = 1, \ldots, M \end{aligned} \tag{4.3}$$

We approximate then (4.2) by

$$H_2(v;v_0,d_1,d_2,T) \cong \left[1 - \frac{[(v_0-\beta d_1-\beta d_2)^2 - v^2\ {}^2]}{4\alpha^2T^2}\right]_+^{1/2} \tag{4.4}$$

$$+ \sum_{i=1}^{m} \left[\left[1 - \frac{(\eta^2_{i-1} - v^2)^2}{4\alpha^2T^2}\right]_+^{1/2} - \left[1 - \frac{(\eta_i{}^2 - v^2)^2}{4\alpha\ 2_T 2}\right]_+^{1/2} \right] \cdot$$

$$\left[1 \ \frac{[(v_0-\beta d_1)^2 - (\tilde{\eta}_i^2 + \beta d_2)^2]^2}{4\alpha^2T^2}\right]_+^{1/2}$$

As M grows (to infinity) the right hand side of (4.4) approaches that of (4.2).

In Table 1 we present the results of some computations of the c.d.f. $H_2(v;v_o,d_1,d_2,T)$ according to approximation (4.4). These computations were performed according to Program BULLET of Appendix 1, with the proper parameters and M = 400 and M = 500. We have tried the approximation also with M = 50, but for small T values (0.1 and 0.2) it has not yielded accurate results for small v values.

4.2 The General Case

After computing the values of $H_2(v;v_o,d_1,d_2,T)$, at specified values of v, one can compute $H_3(v;\ v_o,d_1,d_2,d_3,T)$ at those values of v, etc. The computation is based on the recursive formula

$$H_n\ (v_j\ v_o,\ \underset{\sim}{d}^{(n)},\ T) = \int_{0-}^{v_o-\beta\sum_{j=1}^{n-1} d_j} H_1(v;x,d_n,T)\ dH_{n-1}(x;v_o,\underset{\sim}{d}^{(n-1)},T) \qquad (4.5)$$

$$= H_{n-1}(v+\beta d_n,v_o,\underset{\sim}{d}^{(n-1)},T) + \int_{v+\beta d_n}^{v_o-\beta\sum_{j=1}^{n-1} d_j} H_1(v;x,d_n,T)\ dH_{n-1}(x;v_o,\underset{\sim}{d}^{(n-1)},T)$$

For every $o \leq v \leq v_o-\beta \sum_{j-1}^{n} d_j$,

where $\underset{\sim}{d}^{(n-1)} = (d_1,\ldots,d_{n-1})$, $\underset{\sim}{d}^{(n)} = (\underset{\sim}{d}^{(n-1)},d_n)$.

TABLE 1. The distributions $H_2(v;\, v_0,\, d_1,\, d_2,\, T)$ for $v_0 = 1300$ [m/sec], $\beta = 1.8$, $\alpha = 2{,}812{,}500$ [m/sec^2], $d_1 = 200$, $d_2 = 250$.

	V/T	.1	.2	.3	.4	.5
	0.	0.000000	0.743070	0.966422	0.990392	0.996216
M = 400	100.	0.000000	0.743149	0.967444	0.990665	0.996321
	200.	0.000000	0.750490	0.970347	0.991447	0.996623
	300.	0.000000	0.781469	0.974685	0.992632	0.997082
	400.	0.000000	0.813191	0.979840	0.994068	0.997642
	500.	0.000000	0.864330	0.985156	0.995578	0.998234
	600.	0.000000	0.868422	0.990042	0.996997	0.998795
	700.	0.257541	0.965970	0.994063	0.998188	0.999269
	800.	0.573220	0.983473	0.996987	0.999070	0.999623
	900.	0.792740	0.993628	0.998799	0.999626	0.999848
	1000.	0.969721	0.998369	0.999685	0.999901	0.999960
	1100.	0.997259	0.999835	0.999968	0.999990	0.999996
	1200.	0.999998	1.000000	1.000000	1.000000	1.000000

PROCESSOR USAGE: 102.4 UNITS

	V/T	.1	.2	.3	.4	.5
M = 500	0.	0.000000	0.713346	0.966422	0.990392	0.996216
	100.	0.000000	0.716970	0.967445	0.990665	0.996321
	200.	0.000000	0.731379	0.970347	0.991447	0.996623
	300.	0.000000	0.765288	0.974685	0.992632	0.997082
	400.	0.000000	0.796574	0.979840	0.994068	0.997642
	500.	0.000000	0.841691	0.985156	0.995578	0.998234
	600.	0.000000	0.881748	0.990042	0.996997	0.998795
	700.	0.259115	0.965970	0.994063	0.998188	0.999269
	800.	0.556019	0.983473	0.996987	0.999070	0.999623
	900.	0.794083	0.993628	0.998799	0.999626	0.999848
	1000.	0.969721	0.998369	0.999685	0.999901	0.999960
	1100.	0.997259	0.999835	0.999968	0.999990	0.999996
	1200.	0.999998	1.000000	1.000000	1.000000	1.000000

PROCESSOR USAGE: 123.9 UNITS

Notice that $H_1(v;x,d_n,T) = 1$ for all $o \leq x \leq v + \beta d_n$.

The integral (4.5) is evaluated numerically, for each $n = 2,3,\ldots$ on a constant grid of v values, being $\eta_i = i\Delta$, $i = 0, 1,2,\ldots, M$ where $\Delta = 25$ [m/sec], by a formula similar to (4.4). In Tables 2 and 3 we present the numerical results of computing five exit distributions recursively. The computations were performed according to Program BUL2 given in Appendix 2. The difference between the two examples is in the value of the α coefficient.

5. DETERMINATION OF THE HIT/KILL PROBABILITIES

In Section 2 we introduced the kill probability function $P_K(v_o,R)$. If the bullet goes through n trees on its way to the target, the kill probability, given the last exit velocity $V_n = v_n$ and the vector of distances $\underset{\sim}{d}^{(n)}$, is $P_K(v_n, R-\xi_n-T)$. Accordingly, the kill probability, given $N = n$ and given $\underset{\sim}{d}^{(n)}$ is, for trees of fixed radius T and $n \geq 1$

$$\Psi_K(v_o,n,\underset{\sim}{d}^{(n)},T) = \int_0^{v_o-\beta\sum_{j=1}^{N} d_j} P_K(x,R-\xi_n-T)\,dH_n(x;v_o,\underset{\sim}{d}^{(n)},T). \quad (5.1)$$

Furthermore, since the conditional distribution of the points of location of the trees, $\xi_1 = D_1$, $\xi_2 = D_1+D_2,\ldots,\xi_n = D_1+\ldots,D_n$, given $N = n$, is like that of ordered statistics from a uniform distribution on (0,R), the conditional kill probability, given $N = n$ and T is for $n \geq 1$

TABLE 2. Distributions of Exit Vlocities; v_0 = 1,300 [m/sec], d_1 = 200, d_2 = 150, d_3 = 125, d_4 = 125, d_5 = 200 [m], α = 474,573.75 [m/sec^2], β = .18, T = .3 [m].

v/n	1	2	3	4	5
0.	0.000000	0.000000	0.000000	0.000000	0.000000
25.	0.000000	0.000000	0.000000	0.000000	0.000000
50.	0.000000	0.000000	0.000000	0.000000	0.000000
75.	0.000000	0.000000	0.000000	0.000000	0.000000
100.	0.000000	0.000000	0.060000	0.000000	0.000000
125.	0.000000	0.000000	0.000000	0.000000	0.000000
150.	0.000000	0.000000	0.000000	0.000000	0.000000
175.	0.000000	0.000000	0.000000	0.000000	0.000000
200.	0.000000	0.000000	0.000000	0.000000	0.000000
225.	0.000000	0.000000	0.000000	0.000000	0.000000
250.	0.000000	0.000000	0.000000	0.000000	0.000000
275.	0.000000	0.000000	0.000000	0.000000	0.000000
300.	0.000000	0.000000	0.300000	0.000000	0.000000
325.	0.000000	0.000000	0.000000	0.000000	0.000000
350.	0.000000	0.000000	0.000000	0.000000	0.000000
375.	0.000000	0.000000	0.000000	0.000000	0.000000
400.	0.000000	0.000000	0.000000	0.000000	0.000000
425.	0.000000	0.000000	0.000000	0.000000	0.005033
450.	0.000000	0.000000	0.000000	0.000000	0.009672
475.	0.000000	0.000000	0.000000	0.000000	0.030402
500.	0.000000	0.000000	0.000000	0.000000	0.057226
525.	0.000000	0.000000	0.000000	0.000000	0.082890
550.	0.000000	0.000000	0.000000	0.000000	0.130717
575.	0.000000	0.000000	0.000000	0.000000	0.185662
600.	0.000000	0.000000	0.000000	0.000000	0.244578
625.	0.000000	0.000000	0.000000	0.000000	0.318746
650.	0.000000	0.000000	0.000000	0.000000	0.415488
675.	0.000000	0.000000	0.000000	0.000000	0.510975
700.	0.000000	0.000000	0.000000	0.021790	0.599678
725.	0.000000	0.000000	0.000000	0.072218	0.675214
750.	0.000000	0.000000	0.000000	0.144083	0.757767
775.	0.000000	0.000000	0.000000	0.223350	0.825301
800.	0.000000	0.000000	0.000000	0.318064	0.879620
825.	0.000000	0.000000	0.000000	0.450056	0.920478
850.	0.000000	0.000000	0.000000	0.580120	0.949602
875.	0.000000	0.000000	0.051706	0.696111	0.969486
900.	0.000000	0.000000	0.165134	0.791503	0.983231
925.	0.000000	0.000000	0.308905	0.864775	0.991359
950.	0.000000	0.000000	0.445347	0.916221	0.995854
975.	0.000000	0.000000	0.613175	0.954470	0.998148
1000.	0.000000	0.000000	0.734902	0.977061	0.999238
1025.	0.000000	0.139957	0.858805	0.989348	0.999715
1050.	0.000000	0.394401	0.925631	0.995492	0.999905
1075.	0.000000	0.605108	0.964734	0.998291	0.999973
1100.	0.000000	0.771504	0.985291	0.999433	0.999993
1125.	0.000000	0.889233	0.994716	0.999840	0.999999
1150.	0.256790	0.955309	0.998403	0.999963	1.000000
1175.	0.647181	0.984625	0.999614	0.999994	1.000000
1200.	0.832639	0.995977	0.999936	1.000000	1.000000
1225.	0.940097	0.999407	1.000000	1.000000	1.000000
1250.	0.992331	1.000000	1.000000	1.000000	1.000000
1275.	1.000000	1.000000	1.000000	1.000000	1.000000

```
PROCESSOP USAGE:    20.3 UNITS
OK
```

TABLE 3. Distribution of Exit Velocities; $v_0 = 1{,}300$ [m/sec], $d_1 = 200$, $d_2 = 150$, $d_3 = 125$, $d_4 = 125$, $d_5 = 200$ [m]; $\alpha = 1{,}000{,}000$ [m/sec^2], $\beta = .18$, $T = .3$ [m].

v/n	1	2	3	4	5
0.	0.000000	0.000000	0.248546	0.849444	0.987763
25.	0.000000	0.000000	0.249499	0.849883	0.987313
50.	0.000000	0.000000	0.252296	0.851183	0.987960
75.	0.000000	0.000000	0.256781	0.853297	0.988273
100.	0.000000	0.000000	0.262751	0.856155	0.988645
125.	0.000000	0.000000	0.269995	0.859681	0.989089
150.	0.000000	0.000000	0.278311	0.863787	0.989598
175.	0.000000	0.000000	0.298052	0.870212	0.990161
200.	0.000000	0.000000	0.313829	0.876223	0.990764
225.	0.000000	0.000000	0.329014	0.882340	0.991492
250.	0.000000	0.000000	0.344104	0.888577	0.992193
275.	0.000000	0.000000	0.371997	0.896986	0.992825
300.	0.000000	0.000000	0.393368	0.904323	0.993625
325.	0.000000	0.000000	0.413887	0.911426	0.994328
350.	0.000000	0.000000	0.448591	0.920106	0.994985
375.	0.000000	0.000000	0.473916	0.927415	0.995663
400.	0.000000	0.000000	0.509298	0.935481	0.996258
425.	0.000000	0.000000	0.538865	0.942599	0.996834
450.	0.000000	0.000000	0.575010	0.949899	0.997338
475.	0.000000	0.000000	0.606748	0.956373	0.997804
500.	0.000000	0.000000	0.644192	0.962792	0.998202
525.	0.000000	0.000000	0.676097	0.968315	0.998559
550.	0.000000	0.000000	0.714444	0.973675	0.998854
575.	0.000000	0.000000	0.744889	0.978037	0.999111
600.	0.000000	0.000000	0.781540	0.982317	0.999318
625.	0.000000	0.022092	0.813364	0.985847	0.999487
650.	0.000000	0.057398	0.840945	0.988732	0.999622
675.	0.000000	0.152739	0.868480	0.991342	0.999726
700.	0.000000	0.239789	0.891901	0.993404	0.999806
725.	0.000000	0.310658	0.912011	0.995058	0.999866
750.	0.000000	0.392621	0.930561	0.996400	0.999909
775.	0.000000	0.472353	0.945940	0.997427	0.999940
800.	0.000000	0.543983	0.958375	0.998197	0.999962
825.	0.000000	0.610063	0.969236	0.998778	0.999976
850.	0.000000	0.679238	0.977751	0.999192	0.999986
875.	0.000000	0.741204	0.984292	0.999481	0.999992
900.	0.000000	0.795375	0.989221	0.999677	0.999995
925.	0.000000	0.843923	0.992890	0.999806	0.999998
950.	0.000000	0.887755	0.995441	0.999888	0.999999
975.	0.000000	0.922429	0.997168	0.999938	0.999999
1000.	0.087551	0.947879	0.998306	0.999967	1.000000
1025.	0.410668	0.965621	0.999033	0.999984	1.000000
1050.	0.564656	0.978007	0.999480	0.999993	1.000000
1075.	0.676127	0.986585	0.999741	0.999997	1.000000
1100.	0.763202	0.992334	0.999883	0.999999	1.000000
1125.	0.832881	0.995996	0.999953	1.000000	1.000000
1150.	0.889612	0.998159	0.999984	1.000000	1.000000
1175.	0.932262	0.999302	0.999996	1.000000	1.000000
1200.	0.964843	0.999807	0.999999	1.000000	1.000000
1225.	0.986826	0.999971	1.000000	1.000000	1.000000
1250.	0.998278	1.000000	1.000000	1.000000	1.000000
1275.	1.000000	1.000000	1.000000	1.000000	1.000000

```
PROCESSOR USAGE:    21.0 UNITS
```

$$F_K(v_0,n,T) =$$

$$\frac{n!}{R^n}\int_0^R d\xi_n \int_0^{\xi_n} d\xi_{n-1} \cdots \int_0^{\xi_1} d\xi_1 \, \Psi_K(v_0,n,\ \xi_1-T,\ \xi_2-\xi_1-2T,\ldots,\xi_n-\xi_{n-1}-2T,T) \tag{5.2}$$

where the vector $(\xi_1-T,\ \xi_2-\xi_1-T,\ldots,\ \xi_n-\xi_{n-1}-2T)$ is substituted in (5.1) for $\underset{\sim}{d}^{(n)}$. For $n = 0$ we define $F_k(v_0,\ 0,T) \equiv P_K(v_0,R)$. Finally, the total kill probability is

$$F_K(v_0,T) = e^{-R\mu}\sum_{n=0}^{\infty}\frac{(R\mu)^n}{n!}F_K(v_0,n,T). \tag{5.3}$$

We discuss here two numerical procedures for the estimation of $F_K(v_0,T)$ and their accuracy. The values of the function $P_K(x,\ R-\xi_n-T)$ are given or computed on the same grid of points $\eta_i = i\Delta$, $i = 0,\ldots,25$, as that used for the numerical computation of the functions $H_n(v;v_0,\underset{\sim}{d}^{(n)},T)$. Thus, the function (5.1) is evaluated numerically according to the formula

$$\Psi_K(v_0,n,\underset{\sim}{d}^{(n)},T) \cong \tag{5.4}$$

$$\sum_{i=1}^{m} P_K(\eta_i^2,R-\Psi_n-T)[H_n(\eta_i;v_0,\underset{\sim}{d}^{(n)},T) - H_n(\eta_{i-1};v_0,\underset{\sim}{d}^{(n)},T)],\ n\geq 1.$$

It is much more complicated to evaluate the function $F_K(v_0,n,T)$ numerically. We have introduced here two methods for a Monte Carlo estimation of (5.2).

Method I

For each n, n = 1, 2,..., NP, perform the following Monte Carlo estimation independently. Simulate n uniform random variables on (0,R). Let $\xi_1, \xi_2, \ldots, \xi_n$ be the order statistics of these simulated variable. Compute the function $\Psi_K(v_0, n, \underset{\sim}{d}^{(n)}, T)$ for these values. Repeat this simulation independently NS times and average the resulting $\Psi_k(\bullet)$ values. This average $\hat{F}_K(v_0,n,T)$ is an unbiased estimator of $F_K(v_0,n,T)$. Let S_n^2 be the sample variance of the simulated $\hat{F}_k(\bullet)$ values. An estimator of the variance of $\hat{F}_K(v_0,n,T)$ is thus S_n^2/NS. Finally, the kill probability $\mathcal{F}_K(v_0,T)$ is estimated by

$$\hat{\mathcal{F}}_K(v_0,T) = e^{-\mu R} \sum_{n=0}^{NP} \frac{(\mu R)^n}{n!} \hat{F}_K(v_0,n,T), \tag{5.5}$$

$$\hat{F}_K(v_0,0,T) = F_K(v_0,0,T) = P_K(v_0,R).$$

NP is a sufficiently large integer, so that the sum of the Poisson probabilities, for n larger than NP, is sufficiently small. In the following numerical examples we specified the value of NP = INT $(\mu R + 3\sqrt{\mu R})$, where INT(x) denotes the integer smaller than or equal to x. Let pos(n; μR) denote the Poisson probability of N = n, with parameter μR. The variance of $\mathcal{F}_K(v_0,T)$ is estimated by

$$V\hat{\mathcal{F}}_K(v_0,T) = \sum_{n=0}^{NP} (\text{pos}(n;\mu R))^2 \, S_n^2/NS. \tag{5.6}$$

The square root of (5.6) is the standard-error of the estimate of the kill probability.

Method II

According to the second method we estimate $F_K(v_o,T)$ in one simulation string; repeat this estimation independently NS times and take the average of the individual estimates. More specifically. Starting with $F_K(v_o,0,T) = P_K(v_o,R)$ we set

$$\hat{F}_K^{(i)}(v_o,T) \longleftarrow \text{pos}(0;\mu R) * F_K(v_o,0,T).$$

We then set n = 1 and generate a uniform random variable from (0,R). This value is set equal to d_1 and the function $\Psi_K(v_o,1,d_1,T)$ is computed.

The value of $\hat{F}_K^{(i)}(v_o,T)$ is then changed to $\hat{F}_K^{(i)}(v_o,T) + \text{pos}(1;R\mu)*$ $\Psi_K(v_o,1,d_1,T)$. We set then n = 2, generate a uniform random variable from $(0,R-d_1)$, which is set to be equal to d_2. We then compute $\Psi_K(v_o,2,d_1,d_2.T)$ and set $\hat{F}_K^{(i)}(v_o,T) \longleftarrow \hat{F}_K^{(i)}(v_o,T) + \text{pos}(2;\mu R)* \Psi_K(v_o,2,d_1,d_2,T)$. This simulation algorithm is continued until n = NP. According to this algorithm

$$\hat{F}_K^{(i)}(v_o,T) = \sum_{n=0}^{NP} \text{pos}(n;\mu R)\,\Psi_K(v_o,n,\underset{\sim}{d}^{(n)},T). \tag{5.7}$$

i designates the index of the simulation run, i = 1,...,NS. Finally, $F_K(v_o,T)$ is estimated by

$$\hat{\hat{F}}_K(v_o,T) = \frac{1}{NS} \sum_{i=1}^{NS} \hat{F}_K^{(i)}(v_o,T). \tag{5.8}$$

An estimate of the variance of $\hat{\hat{F}}_K(v_o,T)$ is obtained by computing the sample variance of the $\hat{F}_K^{(i)}(v_o,T)$ values (i = 1,...,NS) and dividing this sample variance by NS. The square-root of this variance is the standard error of $\hat{F}_K(v_o,T)$. In Appendix 3 we provide Program BUL6 which estimates the kill probabilities $F_K(v_o,T)$ according to Method I. Program BUL7 given in Appendix 4 is designed for the estimation of $F_K(v_o,T)$ according to Method II. In both programs we considered the special function

$$P_K(v_o,R) = \begin{cases} 1, \text{ if } v_o-\beta R \geq v_k \\ 0, \text{ otherwise.} \end{cases} \tag{5.9}$$

In Program BUL6 this special $P_K(v_o,R)$ function is programmed in the main routine. Program BUL7 is written in a more general manner. The function $P_K(v_o,R)$ is computed as a subroutine function and can be changed without altering the main program. In Table 4 we provide estimates of the kill probability computed according to Method I and Method II with the function $P_K(v_o,R)$ given by (5.9). For Method I we present the average estimates of $\Psi_K(v_o,n,\underset{\sim}{d}^{(n)},T)$ and the corresponding sample standard deviations of these estimates for n = 0,1,...,NP. The estimate $F_K(v_o,T)$ and its standard error are given, too. For Method II we present the estimates $F_K^{(i)}(v_o,T)$ for i = 1,...,NS, the average $F_K(v_o,T)$ and its standard error. We see that the accuracy of Method I is somewhat higher than that of Method II. However, Method I requires more than 5 times longer computer time than Method II. It seems justified to recommend the use of Method II.

TABLE 4. Estimates of the Kill Probability by Method I and Method II; v_o = 1,300 [m/sec], μ = .005, R = 1,000 [m], α = 474,573.75 [m/sec^2], β = .18, v_k = 500 [m/sec].

METHOD I			METHOD II	
n	$\bar{\Psi}_K(v_o,n,\underline{d}^{(n)},T)$	S_n	i	$\hat{P}_K^{(i)}(v_o, T)$
0	1.000000	0.000000	1	0.646316
1	1.000000	0.000000	2	0.632648
2	1.000000	0.000000	3	0.633372
3	1.000000	0.000000	4	0.651003
4	0.999649	0.001109	5	0.613876
5	0.796708	0.092621	6	0.615645
6	0.295945	0.039027	7	0.604395
7	0.082902	0.019183	8	0.631388
8	0.015948	0.005775	9	0.664867
9	0.001935	0.000375	10	0.692548
10	0.000415	0.000254		
11	0.000035	0.000009		
$\hat{P}_K$ = .633280 S.E.$\{\hat{P}_K\}$ = .005485 Processor Usage: 1977.9 units			$\hat{\hat{P}}_K$ = .638706 S.E.$\{\hat{\hat{P}}_K\}$ = .008206 Processor Usage: 368.5 units	

6. GENERALIZATION FOR RANDOM TREE RADIUS

In the present section we indicate how the previous results can be generalized to the case of random trunk radius, T. Let $F_T(T)$ be the c.d.f. of T. Given that N = n, the corresponding values $T_1,,\ldots T_n$ are independent and identically distributed. The following modifications are needed. Let $\underset{\sim}{T}^{(n)} = (T_1,\ldots,T_n)$. The exit velocity distributions are then computed as

$$H_1(v;v_0,d_1,T_1) \qquad , \text{ for } n = 1 \qquad (6.1)$$

$$H_2(v;v_0,d_1,d_2,T_1,T_2) = \int H_1(v;x,d_2,T_2)dH_1(x;v_0,d_1T_1) \qquad , \text{ for } n = 2 \qquad (6.2)$$

and

$$H_n(v;v_0,\underset{\sim}{d}^{(n)},\underset{\sim}{T}^{(n)}), = \int H_1(v;x,d_n,T_n)dH_{n-1}(x;v_0,\underset{\sim}{d}^{(n-1)},\underset{\sim}{T}^{(n-1)}), \text{ for } n > 2. \qquad (6.3)$$

Given these exit velocity distributions, we determine for each $n \geq 1$, the conditional kill probabilities

$$\Psi_K(v_0,n,\underset{\sim}{d}^{(n)},\underset{\sim}{T}^{(n)}) = \int P_K(x,R-\xi_n-T_n)dH_n(x;v_0,\underset{\sim}{d}^{(n)},\underset{\sim}{T}^{(n)}). \qquad (6.4)$$

The kill probabilities, given v_0, and $\{N=n\}$ are computed then as

$$\bar{F}_K(v_0,n) = \int\cdots\int F_K(v_0,n,\underset{\sim}{t}_n) \prod_{i=1}^{n} dF_T(t_i), \qquad (6.5)$$

where $F_K(v_o, n, \underset{\sim}{t}_n)$ is a generalization of the function defined in (5.2) obtained by integrating $\Psi_K(v_o, n, \xi_1 - T, \xi_2 - \xi_1 - 2T, \ldots, \xi_n - \xi_{n-1} - 2T, \underset{\sim}{t}_n)$ over the simplex of $\xi_1 \leq \xi_2 \leq \ldots \leq \xi_n$. This integral is estimated, as in the previous section by Method I. The integral (6.5) can then be evaluated numerically either by evaluating a discrete version of it, or by simulation. If one employs Method II then (6.5) is evaluated for each n in the same string of computations. Finally, in Method I the functions $\mathcal{F}_K(v_o)$ are computed as Poisson averages of $\bar{F}_K(v_o, n)$.

References

1. Herbert A. David
   Order Statistics
   John Wiley and Sons, Inc., New York, 1970.

2. Edward J. Dudewicz
   Introduction to Statistics and Probability
   Holt, Rinehard and Winston, New York, 1976.

3. J. P. Lambert
   A Residual Velocity Predictive Model for Long Rod Penetrators (U)
   Memorandum Report ARRRL-MR-02828, Ballistic Research Laboratory,
   Aberdeen Proving Ground, Maryland, (CONFIDENTIAL), 1978.

APPENDIX I. FORTRAN PROGRAM BULLET

(Computation of $H_2(v;\ v_o,\ d_1,\ d_2,\ T)$

```
      DIMENSION H(5)
      V0=1300.
      D1=200.
      D2=250.
      A=1750000.
      B=.13
      V1=V0-B*D1
      V2=V1-B*D2
      M=500
      AM=M
      K=13
      DO 1 I=1,K
      AI=I-1
      VI=100.*AI
      DO 20 L=1,5
      AL=L
      T=.1*AL
      WI=1.-((V2*V2-VI*VI)/(2.*A*T))**2
     .IF(WI) 2,3,3
    2 WI=0.
    3 H(L)=SQRT(WI)
      W1S=V1*V1-2.*A*T
      IF(W1S) 40,41,41
   40 W1S=0.
   41 V1S=SQRT(W1S)
      VS=V1S-B*D2
      IF(VI-VS) 42,43,43
   42 RI=V2-VS
      YS=VS
      GO TO 50
   43 RI=V2-VI
      YS=VI
   50 DO 4 J=1,M
      AJ=J
      YJ=YS+AJ*RI/AM
      YJJ=YJ-RI/AM
      YJM=(YJ+YJJ)/2.
      ZJ=((YJ*YJ-VI*VI)/(2.*A*T))**2
      ZJJ=((YJJ*YJJ-VI*VI)/(2.*A*T))**2
      IF(ZJ-1.) 5,4,4
    5 IF(ZJJ-1.) 6,4.4
    6 GJ=SQRT(1.-ZJ)-SQRT(1.-ZJJ)
      UJ=((V1*V1-(YJM+B*D2)**2)/(2.*A*T))**2
      IF(UJ-1.) 7,4,4
    7 TJ=SQRT(1.-UJ)
      H(L)=H(L)-TJ*GJ
    4 CONTINUE
   20 CONTINUE
      PRINT 10,VI,(H(L),L=1,5)
   10 FORMAT(5X,F6.0,5F10.6)
    1 CONTINUE
    . END
```

VO ← v_o

A ← α

B ← β

T ← trunk radius

APPENDIX II. FORTRAN PROGRAM BUL2

(Recursive computation of $H_n(v;\, v_0,\, \underset{\sim}{d}^{(n)},\, T)$)

E ← Δ

A ← α

```
      DIMENSION H(5,52),D(5),U(5)
      DATA (D(I),I=1,5)/200.,150.,125.,125.,200./
      E=25.
      K=5
      V0=1300.
      A=1000000.
      T=.3
      B=.18
      M=52
      AM=M
      V1=V0-D(1)*B
      U(1)=V1
      DO 100 I=1,M
      AI=I-1
      VI=AI*E
      ZI=(V1*V1-VI*VI)/(2.*A*T)
      IF(ZI) 101,101,102
  101 H(1,I)=1.
      GO TO 100
  102 IF(ZI-1.) 103,103,104
  103 H(1,I)=SQRT(1.-ZI*ZI)
      GO TO 100
  104 H(1,I)=0.
  100 CONTINUE
      DO 200 L=2,K
      U(L)=U(L-1)-D(L)*B
      VL=U(L)
      DO 300 I=1,M
      AI=I-1
      VI=AI*E
      IF(VI-VL) 301,301,302
  301 H(L,I)=0.
      DO 400 J=1,M
      AJ=J-1
      XJ=E*(AJ+.5)
      ZJ=(XJ-D(L)*B)*(XJ-D(L)*B)-VI*VI
      ZJ=ZJ/(2.*A*T)
      IF(ZJ) 401,402,402
  401 PIJ=1.
      GO TO 405
  402 IF(ZJ-1.) 403,403,404
  403 PIJ=SQRT(1.-ZJ*ZJ)
      GO TO 405
  404 PIJ=0.
  405 IF(J-1) 406,406,407
  406 H(L,I)=H(L,I)+PIJ*H(L-1,J)
      GO TO 400
  407 H(L,I)=H(L,I)+PIJ*(H(L-1,J)-H(L-1,J-1))
  400 CONTINUE
      GO TO 300
  302 H(L,I)=1.
  300 CONTINUE
  200 CONTINUE
      DO 500 I=1,M
      II=I-1
      AI=II
      VI=AI*E
      PRINT 510,VI,(H(L,I),L=1,K)
  510 FORMAT(5X,F6.0,5F10.6)
  500 CONTINUE
      END
```

APPENDIX III. FORTRAN PROGRAM BUL6

(Estimation of $F_K(v_0, T)$ by Method I)

```
100$LIB,RANDX,,,***
110          DIMENSION G(52),H(52),D(100),U(100)
120          R=1000.
130          V0=1300.
140          A=474573.75
150          AL=5.
160          AMU=R/AL
170          T=.3
180          VK=500.
190          B=.13
200          NR=100
210          NP=11
220.         M=52
230          AM=M
240          NS=10
250          AMS=NS
260          E=25.
270          Y=RAND(-1.)
230          DO 1 I=1,NR
290          Y=RAND(0.)
300        1 CONTINUE
310          K=0
320          FK=POS(K,AL)
330          VR=V0-B*R
340          IF(VR-VK) 21,21,22
350       21 PK=0.
360          GO TO 23
370       22 PK=1.
380       23 TPK=PK*FK
390          QPK=0.
400          PRINT 24,K,PK,QPK
410       24 FORMAT(5X,I4,2F10.6)
420          DO 600 K=1,NP
430          SPK=0.
440          SSPK=0.
450          DO 20 LS=1,NS
460          W=0.
470          Q=0.
430          DO 2 J=1,K
490          Y=RAND(0.)
500          D(J)=Y*(R-W)
510          W=W+D(J)
520          Q=Q+B*D(J)
530          U(J)=Q
```

← Library function for generating uniform M.V.'s on (0,1)

$A \leftarrow \alpha$

$AL \leftarrow R\mu$

This loop is just for generating the first NR = 100 uniform R.V.'s

APPENDIX III (Continued)

```
540      2 CONTINUE
550        IF(K-1) 3,3,9
560      3 V1=V0-U(1)
570        VKK=VK+B*R-U(K)
580        ZK=(V1*V1-VKK*VKK)/(2.*A*T)
590        IF(ZK) 31,32,32
600     31 PK=0.
610        GO TO 35
620     32 IF(ZK-1.) 33,33,34
630     33 PK=1.-SQRT(1.-ZK*ZK)
640        GO TO 35
650     34 PK=1.
660     35 SPK=SPK+PK
670        SSPK=SSPK+PK*PK
680        GO TO 20
690      9 V1=V0-U(1)
700        DO 100 I=1,M
710        AI=I-1
720        VI=E*AI
730        ZI=(V1*V1-VI*VI)/(2.*A*T)
740        IF(ZI) 101,101,102
750    101 G(I)=1.
760        GO TO 100
770    102 IF(ZI-1.)103,103,104
780    103 G(I)=SQRT(1.-ZI*ZI)
790        GO TO 100
800    104 G(I)=0.
810    100 CONTINUE
820        DO 200 L=2,K
830        VL=V0-U(L)
840        DO 300 I=1,M
850        AI=I-1
860        VI=E*AI
870        IF(VI-VL) 301,301,302
880    301 H(I)=0.
890        DO 400 J=1,M
900        AJ=J-1
910        XJ=E*(AJ+.5)
920        ZJ=(XJ-D(L)*B)*(XJ-D(L)*B)-VI*VI
930        ZJ=ZJ/(2.*A*T)
940        IF(ZJ) 401,402,402
950    401 PIJ=1.
960        GO TO 405
970    402 IF(ZJ-1.) 403,403,404
980    403 PIJ=SQRT(1.-ZJ*ZJ)
990        GO TO 405
1000   404 PIJ=0.
1010   405 IF(J-1) 406,406,407
1020   406 H(I)=H(I)+PIJ*G(J)
1030       GO TO 400
1040   407 H(I)=H(I)+PIJ*(G(J)-G(J-1))
1050   400 CONTINUE
1060       GO TO 300
1070   302 H(I)=1.
1080   300 CONTINUE
1090       DO 500 I=1,M
1100       G(I)=H(I)
1110   500 CONTINUE
1120   200 CONTINUE
```

APPENDIX III (Continued)

```
1130        VKK=VK+B*R-U(K)
1140        IK=INT(VKK/E)+1
1150        PK=1.-(H(IK)+H(IK-1))/2.
1160        SPK=SPK+PK
1170        SSPK=SSPK+PK*PK
1180     20 CONTINUE
1190        APK=SPK/ANS
1200        SDK=(ANS*SSPK-SPK*SPK)/ANS
1210        SPK=SQRT(SDK/(ANS-1.))
1220        PRINT 24,K,APK,SPK
1230        FK=POS(K,AL)-POS(K-1,AL)
1240        TPK=TPK+FK*APK
1250        QPK=QPK+FK*FK*SPK*SPK/ANS
1260    600 CONTINUE
1270        SD=SQRT(QPK)
1280        PRINT 61,TPK,SD
1290     61 FORMAT(//,5X,7HES. PK=,F10.6,7HES. SD=,F10.6)
1300        END
```

Subroutine function for computing the Poisson c.d.f.

```
1310        FUNCTION POS(J,AL)
1320        I=J
1330         B=AL
1340        IF(B.GE.10.) GO TO 8
1350        IF(I) 1,2,3
1360      1 POS=0.
1370        GO TO 10
1380      2 POS=EXP(-B)
1390        GO TO 10
1400      3 POS=EXP(-B)
1410        F=POS
1420        DO 4 K=1,I
1430        AK=K
1440        F=F*B/AK
1450        POS=POS+F
1460      4 CONTINUE
1470        GO TO 10
1480      8 AI=I+.5
1490        ZI=(AI-B)/SQRT(B)
1500        POS=CNDX(ZI)
1510     10 RETURN
1520        END
```

For large AL it needs the normal c.d.f.

Subroutine function for computing the standard normal c.d.f.

```
1530        FUNCTION CNDX(X)
1540        Y=X
1550        ISWTCH=0
1560        IF(Y) 1,2,2
1570      1 Y=ABS(Y)
1580        ISWTCH=1
1590      2 P=.2316419
1600        B1=.31938153
1610        B2=-.35656378
1620        B3=1.7814779
1630        B4=-1.8212559
1640        B5=1.3302744
1650        T=1./(1.+P*Y)
1660        R=.3989423*EXP(-Y*Y/2.)
1670        QNDX=1.-R*(B1*T+B2*T*T+B3*T*T*T+B4*(T**4)+B5*(T**5))
1680        IF(ISWTCH) 3,4,3
1690      3 QNDX=1.-QNDX
1700      4 CNDX=QNDX
1710        RETURN
1720        END
```

APPENDIX IV. FORTRAN PROGRAM BUL7

(Computing $F_K(v_0, T)$ according to Method II)

```
$LIB,RANDX,..***
       DIMENSION G(52),H(52),D(100),U(100)
       R=1000.
      BL=.083
       V0=1300.
       A=474573.75
       AL=BL*R
      T=.15
       VK=500.
       B=.18
       NR=100
       NP=INT(AL+3.*SQRT(AL))
       M=52
       AM=M
       NS=10
       ANS=NS
       E=25.
       Y=RAND(-1.)
       DO 1 I=1,NR
       Y=RAND(0.)
     1 CONTINUE
       K=0
       PK=FK(V0,R)
       TPK=PK*POS(K,AL)
       WPK=TPK
       QPK=0.
       SPK=0.
       DO 20 LS=1,NS
       TPK=WPK
       W=0.
       Q=0.
       K=1
       Y=RAND(0.)
       D(1)=Y*(R-W)
       W=W+D(1)
       Q=Q+B*D(1)
       U(1)=Q
       V1=V0-U(1)
       PK=0.
       DO 100 I=1,M
       AI=I-1
       VI=E*AI
       ZI=(V1*V1-VI*VI)/(2.*A*T)
       IF(ZI) 101,101,102
   101 G(I)=1.
       GO TO 105
   102 IF(ZI-1.)103,103,104
   103 G(I)=SQRT(1.-ZI*ZI)
       GO TO 105
   104 G(I)=0.
   105 VIT=VI+E/2.
       IF(I.GT.1) GO TO 106
       PK=PK+FK(VIT,R-W)*G(1)
       GO TO 100
```

Library function for generating uniform R.V.'s on (0,1).

BL ← μ

AL ← μR

APPENDIX IV (Continued)

```
 106 PK=PK+FK(VIT,R-W)*(G(I)-G(I-1))
 100 CONTINUE
     TPK=TPK+PK*(POS(K,AL)-POS(K-1,AL))
     DO 200 L=2,NP
     Y=RAND(0.)
     D(L)=Y*(R-W)
     W=W+D(L)
     Q=Q+B*D(L)
     U(L)=Q
     VL=V0-U(L)
     PK=0.
     DO 300 I=1,M
     AI=I-1
     VI=E*AI
     VIT=VI+E/2.
     IF(VI-VL) 301,301,302
 301 H(I)=0.
     DO 400 J=1,M
     AJ=J-1
     XJ=E*(AJ+.5)
     ZJ=(XJ-D(L)*B)*(XJ-D(L)*B)-VI*VI
     ZJ=ZJ/(2.*A*T)
     IF(ZJ) 401,402,402
 401 PIJ=1.
     GO TO 405
 402 IF(ZJ-1.) 403,403,404
 403 PIJ=SQRT(1.-ZJ*ZJ)
     GO TO 405
 404 PIJ=0.
 405 IF(J-1) 406,406,407
 406 H(I)=H(I)+PIJ*G(J)
     GO TO 400
 407 H(I)=H(I)+PIJ*(G(J)-G(J-1))
 400 CONTINUE
     GO TO 303
 302 H(I)=1.
 303 RK=FK(VIT,R-W)
 306 IF(I.GT.1) GO TO 304
     PK=PK+RK*H(1)
     GO TO 300
 304 PK=PK+RK*(H(I)-H(I-1))
 300 CONTINUE
     DO 500 I=1,M
     G(I)=H(I)
 500 CONTINUE
     TPK=TPK+PK*(POS(L,AL)-POS(L-1,AL))
 200 CONTINUE
     SPK=SPK+TPK
     QPK=QPK+TPK*TPK
     PRINT 24,LS,TPK
  24 FORMAT(5X,I4,F10.6)
  20 CONTINUE
     APK=SPK/ANS
     VPK=(ANS*QPK-SPK*SPK)/ANS
     SDK=SQRT(VPK/(ANS-1.)*ANS)
     PRINT 61,APK,SDK
  61 FORMAT(5X,F10.6,5X,F10.6)
     END
```

Subroutine function for computing the Poisson c.d.f.

```
      FUNCTION POS(J,AL)
      I=J
      B=AL
      IF(B.GE.10.) GO TO 8
      IF(I) 1,2,3
    1 POS=0.
      GO TO 10
    2 POS=EXP(-B)
      GO TO 10
    3 POS=EXP(-B)
      F=POS
      DO 4 K=1,I
      AK=K
      F=F*B/AK
      POS=POS+F
    4 CONTINUE
      GO TO 10
    8 AI=I+.5
      ZI=(AI-B)/SQRT(B)
      POS=CNDX(ZI)
   10 RETURN
      END
```

Subroutine function for computing the standard normal c.d.f.

```
      FUNCTION CNDX(X)
      Y=X
      ISWTCH=0
      IF(Y) 1,2,2
    1 Y=ABS(Y)
      ISWTCH=1
    2 P=.2316419
      B1=.31938153
      B2=-.35656378
      B3=1.7814779
      B4=-1.8212559
      B5=1.3302744
      T=1./(1.+P*Y)
      R=.3989423*EXP(-Y*Y/2.)
      QNDX=1.-R*(B1*T+B2*T*T+B3*T*T*T+B4*(T**4)+B5*(T**5))
      IF(ISWTCH) 3,4,3
    3 QNDX=1.-QNDX
    4 CNDX=QNDX
      RETURN
      END
```

Subroutine function for computing $P_K(v_o, R)$

```
       FUNCTION FK(V,R)
      W=V
      P=R
      WK=500.
      B=.18
       WS=W-P*B
      IF(WS-WK) 1,1,2
    1 FK=0.
      GO TO 10
    2 FK=1.
   10 RETURN
      END
```

A COMPARISON OF VARIOUS METHODS FOR COMPUTING INSTANTANEOUS IMPACT PREDICTIONS OF MISSILES FOR RANGE FLIGHT SAFETY

Jerry F. Kuzanek
Mathematical Services Branch
Analysis and Computation Division
National Range Operations Directorate
White Sands Missile Range, New Mexico 88002

ABSTRACT. Three distinct sets of ordinary differential equations, describing the motion of a missile subjected to gravity and drag forces, are developed using Cowell's method, Encke's method, and the method of variation of parameters. These equations are then numerically integrated using various methods such as Adams-Moulton, Runge-Kutta, rational extrapolation, and Gauss-Jackson (Σ^2), in order to compute the missile's trajectory. The intersection of these trajectories with the earth's surface produces several instantaneous impact predictions (IIPs) which are compared for various drag coefficients and error criteria, in an attempt to maximize the accuracy of these computed IIPs while minimizing the computational time required. Furthermore, several techniques for significantly improving the accuracy for approximating drag-corrected IIPs between computational cycles are developed. Finally, other methods for reducing the computational time required to compute drag-corrected IIPs, which have not yet been implemented, are discussed.

1. INTRODUCTION.

During missile flight tests at White Sands Missile Range (WSMR), data on the current status of the missile, such as its position and velocity, is transmitted 20 times per second from radar sites to a Univac 1108 computer. Consecutive pairs of such data are averaged 10 times per second and computations for plotting displays such as current position, range verses altitude, or impact prediction are based upon this averaged data.

In the event that a missile veers off course from its planned trajectory, it will be necessary to terminate thrust to prevent the missile from impacting in a populated area. For this reason, Range Flight Safety requires that for each computational cycle (10 per second) an instantaneous impact prediction (IIP) of the missile be computed. This point is the intersection of the missile trajectory, should thrust be terminated, with the Clarke Spheroid (of 1866) model of the earth at an altitude of 4000 feet.

If the effects of atmospheric drag are neglected then a vacuum IIP can be rapidly computed using Kepler's central force equations, since the missile trajectory is an ellipse. This method is applicable for a variety of low drag vehicles whose exit from and re-entry to the earth's atmosphere occur at high angles. In addition, the vacuum IIP represents an upper limit for the drag-corrected IIP. Since approximately one millisecond is required to compute a vacuum IIP, it is always computed at a 10 per second rate. Details of this computation are given in [1].

For low altitude missiles, whose trajectory is significantly affected by atmospheric drag, a drag-corrected IIP calculation is also required by Range Flight Safety. Since the ordinary differential equations describing the missile trajectory in the atmosphere are not solvable in closed form, as they were in the case of an elliptical trajectory for a missile in a vacuum, these equations must be numerically integrated until the resulting trajectory pierces the earth's surface. The numerical method currently being used is a fourth-order Runge-Kutta (RK) method with approximately ten steps per trajectory length or a second-order RK method with approximately twenty steps per trajectory length. The fourth-order RK method is used whenever the theoretical weight-to-drag ratio (Beta) of the missile is greater than 200, whereas the second-order RK method is used whenever Beta is less than or equal to 200. In either case, approximately forty evaluations of the equations of motion are required per drag IIP computation. Since the Runge-Kutta method is a single step method, the step size at each integration step is independent of the step size at all previous steps. The step size used is adjusted at each step so that it decreases as drag increases, in order to obtain a more accurate drag IIP in the same number of integration steps. A derivation of the equations of motion and the method of solution is described in [2], whereas the step size adjustment is described in [3].

2. STATEMENT OF PROBLEM. Experimental runs have indicated that for some missiles, only one drag IIP could be computed at a ten per second rate and at most four drag IIPs at a five per second rate [4]. Until recently drag IIPs were computed at a five per second rate, with a linear extrapolation of the last two computed drag IIPs being used to approximate the drag IIP at the next intermediate time. Additional mission requirements have necessitated the implementation of a variable rate for computing drag IIPs, ranging from ten per second to two per second [4]. At a two per second rate, four drag IIP approximations via linear extrapolation are required for each computed drag IIP, in order to output IIPs at a ten per second rate. Consequently, the output drag IIPs at a two per second rate are not as smooth and accurate as they are at a five or ten per second rate.

The purpose of this paper is to investigate alternative methods of computing drag IIPs and alternative methods of obtaining approximations to the drag IIPs at intermediate times, in order to decrease computation time and/or increase accuracy. It is a condensed version of a much more detailed report [35]. In order to put this paper in its proper perspective, we briefly mention some previous investigations toward achieving these goals.

3. PREVIOUS INVESTIGATIONS. In 1965 a method for obtaining a drag-corrected Kepler IIP for Athena missiles was investigated [5]. Briefly this method consisted of using position and velocity data from a previous Athena mission to calculate, in non-real-time, a table of differences between Kepler IIPs and drag-corrected IIPs as a function of velocity. During the next Athena mission a linear interpolation of this data was used to obtain a drag-corrected Kepler IIP in real-time. In order for this method to be most effective, the trajectories of the two Athena missiles need to be similar. However, if this were the case, then there would be no need to terminate thrust of the missile, since thrust needs to be terminated only when the missile deviates radically from its expected trajectory. In other words, drag-corrected IIPs should be based upon actual missile positions and velocities rather than upon expected ones.

In 1975 various numerical integration methods for computing drag IIPs using digital and/or analog computers were investigated [6]. The equations of motion which were integrated are those which describe the actual position and velocity of the missile (Cowell's method) in an earth centered fixed (ECF), (non-inertial) cartesian coordinate system. The numerical integration methods investigated included Adams-Moulton, Milne-Hamming, Euler, a variable order Adams method called DIFSUB developed by C. W. Gear [7, 8, 9], and 2nd, 3rd, and 4th order Runge-Kutta methods. The conclusions reached were that the RK methods were better than the other methods, except possibly the method of Gear. Drag IIPs from the various methods were compared against drag IIPs from a 4th order RK method with a step size of 0.5 seconds. Large errors were associated with the analog solutions. The RK method was improved by using a variable step size as a function of drag and by switching to a Kepler IIP whenever the missile altitude exceeds 200,000 feet. Both of these features are presently being utilized.

4. NEWER NUMERICAL INTEGRATION METHODS. Newer, more efficient numerical integration methods have been developed recently which were not considered above. Among these are a fourth-order Runge-Kutta method which minimizes truncation errors [25, p. 200], a rational extrapolation method [10, 11, 26, 27], an improved variable order, variable step Adams method [12], and a Gauss-Jackson (Σ^2) method [13].

Two recent studies [14, 15], comparing the rational extrapolation method, the variable order Adams methods and the Runge-Kutta methods for different sets of test problems, concluded that in general the rational extrapolation method performed best, whereas a later study [16] concluded that in general the variable order Adams codes, such as STEP [12] or DVDQ [17], performed best. All studies were careful to point out that much depended upon the nature of the differential equations being solved, the amount of computation required for function evaluations, and the amount of accuracy required. In general, the variable order Adams methods are best when seeking maximum accuracy or if the equations are expensive to evaluate; the rational extrapolation method is best if the equations are cheap to evaluate, high accuracy is desired, and the solution is reasonably smooth; and the RK methods are best if the equations are cheap to evaluate, low to medium accuracy is desired, and/or storage is at a premium [16, p. 405]. At a recent Society for Industrial and Applied Mathematics (SIAM) meeting [18], Larry Shampine suggested to me that the rational extrapolation method might be best for obtaining a reasonably accurate drag IIP in a given amount of computer time. Since he is also the author of the very efficient variable order Adams code known as STEP, it was decided to code the rational extrapolation method rather than STEP.

In order to apply the variable order, variable step Adams methods, the rational extrapolation method, and the RK methods to the three second-order equations of motion, these equations must first be reduced to a set of six first-order ordinary differential equations. On the other hand, the Gauss-Jackson (Σ^2) method uses a finite difference table to numerically integrate a system of second-order differential equations directly, without the need to reduce such a system to an equivalent first-order system. It has been used successfully in the determination of orbits of minor planets, comets, and satellites. One of the few thorough discussions of this method is contained in Vol. 2 of Herrick's book on Astrodynamics [13, pp. 12-16 and pp. 245-304]. Some of his comments, which encouraged the implementation of this Gaussian method for computing missile trajectories, are the following.

Herrick states that "The present study indicates that for moderate-eccentricity orbits the Gaussian formulae are fifty times more effective than the commonly used fourth-order Runge-Kutta formula" [13, p. 1]. He further states [13, p. 12] that "The Σ^2 procedure, sometimes called 'Gauss-Jackson' ... accomplishes with a 'predictor' formula alone what the δ^2 procedure and related procedures, such as Adams-Bashforth, require 'predictor' and 'corrector' cycles to accomplish." "Only if the latter methods require a 'second-corrector' ... would the Σ^2 procedure be likely to require a 'first-corrector', and such calculations would be ones attempting integration steps that should be shortened." In addition "In Gaussian numerical integration the calculus of finite differences and numerical analysis achieve the height of excellence. ...To the best of my knowledge there has been no prior thorough going study of Gaussian integration", [13, p. 245].

5. EQUATIONS OF MOTION USING COWELL'S METHOD. Cowell's method involves the numerical integration of the total acceleration by any numerical process (e.g. Adams-Moulton, Runge-Kutta, etc.) and usually in rectangular coordinates, i.e., the numerical integration of

$$(5.1) \qquad \ddot{\bar{r}} = -\frac{\mu\bar{r}}{r^3} + \overset{\cdot\,`}{\bar{r}}$$

where $\bar{r} = (x, y, z)$ is the position vector of the missile, $\ddot{\bar{r}}$ its acceleration vector, $r = (x^2 + y^2 + z^2)^{\frac{1}{2}}$ the distance of the missile from the origin of an inertial rectangular coordinate system, μ is a gravitational constant, and $\overset{\cdot\,`}{\bar{r}}$ is the perturbing acceleration. If r is measured in feet and time in seconds, then $\mu = 1.406559714 \times 10^6$ ft^3/sec^2.

The coordinate system used is right-handed, with origin at the earth's center (geocentric), x-y plane in the equatorial plane, z-axis positive through the South Pole, and the x-axis aligned at 106°20' west longitude at the instant missile position and velocity data is obtained. Since this data is obtained in a non-inertial (relative) coordinate system which rotates with the earth, the relative velocity components $\dot{x}_r$, $\dot{y}_r$ must be convereted to corresponding inertial velocity components $\dot{x}$, $\dot{y}$ by

$$(5.2) \qquad \left\{ \begin{array}{l} \dot{x} = \dot{x}_r + \omega y \\ \dot{y} = \dot{y}_r - \omega x \end{array} \right.$$

where $\omega = 7.29211583 \times 10^{-5}$ rad/sec is the earth's angular rate of rotation.

The perturbing acceleration $\overset{\cdot\,`}{\bar{r}}$ can include terms to account for the effects of gravitational anomolies (due to the earth's oblateness or to the missile being near mountains or valleys), wind, lift, drag, and any number of other factors.

However, for real-time IIP computations, the effects of drag are by far the most significant and are therefore the only ones considered here. In an inertial coordinate system, we have

$$(5.3) \qquad \dot{\bar{r}} = -\frac{1}{2}\frac{\rho(h)\,\dot{r}_r}{\beta}\,\dot{\bar{r}}_r,$$

where

$$(5.4) \qquad \dot{\bar{r}}_r = \dot{\bar{r}} + \bar{r}_r \times \bar{\omega}$$

is the relative velocity of the missile,

$$(5.5) \qquad \bar{\omega} = (0,\ 0,\ -\omega),$$

β is a theoretical weight-to-drag ratio of the missile, and $\rho(h)$ is the density of the atmosphere at an altitude of h feet above sea level. We have,

$$(5.6) \qquad h = r - R + 4000$$

where R is the earth radius of the Clarke Spheroid of 1866 at the same latitude of the missile and at an altitude of 4000 feet. We use (cf. [1, p. 4])

$$(5.7) \qquad R = 20929831.0 - 71303.68411\,\frac{z^2}{r^2}.$$

Once the coordinates and time T to impact in an inertial frame have been determined, the coordinates of the true IIP can be obtained by rotating the earth through ωT radians.

Integrating the equations of motion in an inertial frame is used in the Kepler method, Encke method, and the method of variation of parameters. However, for Cowell's method it is actually more convenient to integrate in an earth centered fixed (ECF) cartesian coordinate system which rotates with the earth. Rearranging (5.4), differentiating, and applying the law of Coriolis [23, p. 60], we obtain

$$(5.8) \qquad \ddot{\bar{r}} = \ddot{\bar{r}}_r - 2\,\dot{\bar{r}}_r \times \bar{\omega} + (\,\bar{r}_r \times \bar{\omega}\,) \times \bar{\omega}.$$

Combining (5.1), (5.3), and (5.8), we obtain the Cowell equations of motion in the relative ECF cartesian coordinate system

$$(5.9) \qquad \ddot{\bar{r}}_r = -\frac{\mu\bar{r}}{r^3} - \frac{1}{2}\,\frac{\rho\dot{r}_r}{\beta}\,\dot{\bar{r}}_r + 2\dot{\bar{r}}_r \times \bar{\omega} - (\bar{r} \times \bar{\omega}) \times \bar{\omega}$$

where the subscript r refers to a quantity in the relative (non-inertial) coordinate system, and its absence refers to a quantity in the inertial system. This equation can be written as a set of six first-order equations in the usual way by considering the components of the relative velocity vector $\dot{\bar{r}}_r$, $\dot{x}_r$, $\dot{y}_r$, and $\dot{z}_r$, to be three new variables.

6. <u>EQUATIONS OF MOTION USING ENCKE'S METHOD</u>. Encke's method involves the numerical integration of the deviation $\ddot{\bar{d}}$ of the inertial acceleration vector from the acceleration vector $\ddot{\bar{r}}_e$ for some easily computable Encke reference (or auxiliary) trajectory, i.e.

$$(6.1) \qquad \ddot{\bar{d}} = \ddot{\bar{r}} - \ddot{\bar{r}}_e,$$

where $\ddot{\bar{r}}$ is the true acceleration vector of the missile in an inertial system. We have [13, p. 22]

$$(6.2) \qquad \ddot{\bar{d}} = \mu\left(\frac{\bar{r}_e}{r_e^3} - \frac{\bar{r}}{r^3}\right) + \dot{\grave{\bar{r}}} - \dot{\grave{\bar{r}}}_e = \frac{\mu}{r_e^3}\,(fq\bar{r} - \bar{d}) + \dot{\grave{\bar{r}}} - \dot{\grave{\bar{r}}}_e,$$

where $\bar{r}_e$ is the position in the Encke reference trajectory, r_e is its magnitude, $\dot{\grave{\bar{r}}}_e$ is the perturbing acceleration in the reference trajectory

$$(6.3) \qquad 2q = \bar{d} \cdot (\bar{r}_e + \bar{r}) / r_e^2,$$

and

$$(6.4) \qquad fq = \frac{2q}{1+2q}\left[1 + \frac{1}{1 + 2q + \sqrt{1 + 2q}}\right].$$

The reference trajectory used in this investigation is the Kepler or vacuum trajectory, so that $\dot{\grave{\bar{r}}}_e = \bar{0}$.

The development of Encke's method with universal variables and f-and-g elliptic formula is from [13, pp. 29-35]. All quantities are expressed in an inertial, geocentric, cartesian coordinate system. An alternate development can be found in [19, pp. 232-234], whereas techniques for improving Encke's method for artificial satellites can be found in [21, 22].

Given initial position $\bar{r}_o$ and velocity $\dot{\bar{r}}_o$ vectors at time t_o, we compute

(6.5) $$r_o = (\bar{r}_o \cdot \bar{r}_o)^{1/2} \qquad D_o = (\bar{r}_o \cdot \dot{\bar{r}}_o) / \sqrt{\mu}$$

(6.6) $$c_o = \frac{r_o(\dot{\bar{r}}_o \cdot \dot{\bar{r}}_o)}{\mu} - 1 \qquad \alpha = (c_o - 1) / r_o$$

If we wish to determine the position and velocity of the missile at time t_i, then we perform a Newton-Raphson iteration to obtain a root of

(6.7) $$F(\hat{X}_i) = \sqrt{\mu}\,(t_i - t_o) - r_o\hat{X}_i - D_o\hat{C}_i - c_o\hat{U}_i = 0,$$

where

(6.8) $$\hat{C}_i = \frac{\hat{X}_i^2}{2!} + \alpha\frac{\hat{X}_i^4}{4!} + \alpha^2\frac{\hat{X}_i^6}{6!} + \alpha^3\frac{\hat{X}_i^8}{8!} + \ldots$$

(6.9) $$\hat{U}_i = \frac{\hat{X}_i^3}{3!} + \alpha\frac{\hat{X}_i^5}{5!} + \alpha^2\frac{\hat{X}_i^7}{7!} + \alpha^3\frac{\hat{X}_i^9}{9!} + \ldots$$

The Newton-Rapson iteration scheme has the form

(6.10) $$X_{n+1} = X_n - \frac{F(X_n)}{F'(X_n)},$$

where the subscript n now refers to the nth iteration of $\hat{X}_i$ at time t_i.

To start the iteration at time t_i, we choose X_n for n=0 to be the converged to value of $\hat{X}_{i-1}$ during the previous iteration, or $X_o = 0$ if this is the first iteration. In (6.10), $F'(X_n)$ has the form

(6.11) $$F'(\hat{X}_i) = -r_o - D_o(\hat{X}_i + \alpha\hat{U}_i) - c_o\,\hat{C}_i\,.$$

We terminate the iteration whenever

(6.12) $$\left|\frac{F(\hat{X}_i)}{F'(\hat{X}_i)}\right| \leq \varepsilon(|\hat{X}_i| + 1),$$

where ε is some number selected by the user. In practice, for $\varepsilon = 10^{-5}$, only a few iterations are required for convergence. If $a = -1/\alpha$ denotes the semi-major axis for an elliptic trajectory in a vacuum and E_o the eccentric anomaly at initial time t_o, then

$$(6.13) \qquad \hat{X}_i = \sqrt{a}\,(E_i - E_o)$$

where E_i is the eccentric anomaly at time t_i.

Next we compute the following

$$(6.14) \qquad \hat{S}_i = \hat{X}_i + \alpha\hat{U}_i, \qquad r_{ei} = r_o + D_o\,\hat{S}_i + c_o\,\hat{C}_i,$$

$$(6.15) \qquad f_i = 1 - \hat{C}_i/r_o, \qquad g_i = t_i - \hat{U}_i \,/\, \sqrt{\mu}\ ,$$

$$(6.16) \qquad \dot{f}_i = -\sqrt{\mu}\ \hat{S}_i \,/\, r_{ei} r_o, \qquad \dot{g}_i = 1 - \hat{C}_i \,/\, r_{ei}\ .$$

The position and velocity vectors of the missile in a vacuum at time t_i are respectively

$$(6.17) \qquad \overline{r}_{ei} = f_i\,\overline{r}_o + g_i\,\dot{\overline{r}}_o\ , \qquad \dot{\overline{r}}_{ei} = \dot{f}_i\,\overline{r}_o + \dot{g}_i\,\dot{\overline{r}}_o\ .$$

The actual position and velocity vectors of the missile are given by

$$(6.18) \qquad \overline{r}_i = \overline{r}_{ei} + \overline{d}_i\ , \qquad \dot{\overline{r}}_i + \dot{\overline{r}}_{ei} + \dot{\overline{d}}_i\ .$$

One of the numerical integration methods is then applied to eq. (6.2), with $\ddot{\overline{r}}_{ei} = \overline{0}$ and

$$(6.19) \qquad \ddot{\overline{r}}_i = -\frac{1}{2}\,\frac{\rho(h_i)\ |\ \dot{\overline{r}}_i + \overline{r}_i \text{ X } \overline{\omega}\ |}{\beta}\,(\dot{\overline{r}}_i + \overline{r}_i \text{ X } \overline{\omega}\,)$$

where h_i is given by (5.6).

7. EQUATIONS OF MOTION USING THE METHOD OF VARIATION OF PARAMETERS.

The method of variation of parameters treats the initial position and velocity vectors of the missile as slowly varying parameters of time, rather than as constant vectors. A solution of (5.1) at time t_i is sought in the form of

$$(7.1) \qquad \overline{r}_i = \overline{r}\,(t_i) = f_i\,\overline{r}_o\,(t_i) + g_i\,\dot{\overline{r}}_o\,(t_i)$$

$$(7.2) \qquad \dot{\bar{r}}_i = \dot{\bar{r}}\,(t_i) = \dot{f}_i\,\bar{r}_o\,(t_i) + \dot{g}_i\,\dot{\bar{r}}_o\,(t_i)\,.$$

If $\grave{\dot{\bar{r}}} \equiv 0$, then $\bar{r}_o\,(t_i) \equiv \bar{r}_o$ and $\dot{\bar{r}}_o\,(t_i) = \dot{\bar{r}}_o$ are constant vectors and (7.1) and (7.2) form the classical f-and-g function solution of the Kepler two-body equations of motion, where f and g functions and their derivatives are given by (6.15) - (6.16).

If $\grave{\dot{\bar{r}}} \neq 0$, then $\bar{r}_o$ (t) and $\dot{\bar{r}}_o$ (t) satisfy [20, p. 6 or 13, p. 119]

$$(7.3) \qquad \grave{\bar{r}}_o = g\,\grave{\dot{\bar{r}}} - \grave{g}\,\dot{\bar{r}} - g\,\grave{\dot{\bar{r}}}\;, \qquad \grave{\dot{\bar{r}}}_o = -\grave{\dot{f}}\;\bar{r} + \grave{f}\,\dot{\bar{r}} + f\,\grave{\dot{\bar{r}}}\;.$$

where the superscript ` denotes a "perturbation time derivative", i.e. for any time dependent function s(t), the total time derivative is

$$(7.4) \qquad \frac{ds}{dt} = \dot{s} + \grave{s}\;,$$

where $\dot{s}$ is the time derivative when the net perturbing acceleration $\grave{\dot{\bar{r}}}$ is zero.

The perturbation derivatives $\grave{f}$, $\grave{g}$, $\grave{\dot{f}}$, and $\grave{\dot{g}}$ appearing in (7.3) can be determined several different ways. One such method uses the universal variables of Herrick [13, pp. 116-120], which requires the computation of the following terms.

$$(7.5) \qquad \begin{cases} r_o = \sqrt{\bar{r}_o \cdot \bar{r}_o}\;, & v_o^2 = \dot{\bar{r}}_o \cdot \dot{\bar{r}}_o\;, \\ D_o = \bar{r}_o \cdot \dot{\bar{r}}_o \Big/ \sqrt{\mu}\;. & \end{cases}$$

$$(7.6) \qquad \alpha = v_o^2 \Big/ \mu - 2 \Big/ r_o\;, \qquad c_o = 1 + r_o\,\alpha\;.$$

As in the Encke method using universal variables, a Newton-Raphson iteration of

$$(7.7) \qquad F(\hat{X}) = \sqrt{\mu}\,(t-t_o) - r_o\,\hat{X} - D_o\,\hat{C} - c_o\,\hat{U} = 0$$

is required, where

$$(7.8) \qquad \begin{cases} \hat{U} = \dfrac{\hat{X}^3}{3!} + \alpha\hat{U}^*\;, & U^* = \dfrac{\hat{X}^5}{5!} + \alpha\,\dfrac{\hat{X}^7}{7!} + \alpha^2\,\dfrac{\hat{X}^9}{9!} + \cdots\;, \\ \hat{C} = \dfrac{\hat{X}^2}{2!} + \alpha\,\hat{C}^*\;, & C^* = \dfrac{\hat{X}^4}{4!} + \alpha\,\dfrac{\hat{X}^6}{6!} + \alpha^2\,\dfrac{\hat{X}^8}{8!} + \cdots\;. \end{cases}$$

Also, the equations (6.14-6.17) are required and are repeated here in slightly different form

$$(7.9)\qquad \hat{S} = \hat{X} + \alpha\hat{U} \qquad\qquad r = r_o + D_o\hat{S} + c_o\hat{C}$$

$$(7.10)\quad \begin{cases} f = 1 - \hat{C}/r_o & g = (t-t_o) - \hat{U}/\sqrt{\mu} \\ \dot{f} = -\sqrt{\mu}\hat{S}/rr_o & \dot{g} = 1 - \hat{C}/r \end{cases}$$

$$(7.11)\qquad \bar{r} = f\,\bar{r}_o + g\,\dot{\bar{r}}_o \qquad\qquad \dot{\bar{r}} = \dot{f}\,\bar{r}_o + \dot{g}\,\dot{\bar{r}}_o$$

In addition, we require the following perturbations [13, p. 119]:

$$(7.12)\qquad D = (\bar{r}\cdot\dot{\bar{r}})/\sqrt{\mu} \qquad\qquad c = 1 + r\,\alpha$$

$$(7.13)\qquad \hat{U}_\alpha = \frac{1}{2}(\hat{X}\hat{C}^* - 3\,\hat{U}^*) \qquad\qquad \hat{C}_\alpha = \frac{1}{2}(\hat{X}\,\hat{U} - 2\,\hat{C}^*)$$

$$(7.14)\qquad \grave{D} = (\bar{r}\cdot\grave{\dot{\bar{r}}})/\sqrt{\mu} \qquad\qquad \grave{\alpha} = 2(\dot{\bar{r}}\cdot\grave{\dot{\bar{r}}})/\mu$$

$$(7.15)\qquad \hat{M}_\alpha\cdot = r\,\hat{U} + c\hat{U}_\alpha - D\,\hat{C}_\alpha \qquad\qquad \hat{X} = (\hat{C}\grave{D} - M_\alpha\,\grave{\alpha})/r_o$$

$$(7.16)\qquad \grave{\hat{U}} = \hat{C}\,\grave{\hat{X}} + \hat{U}_\alpha\grave{\alpha} \qquad\qquad \grave{\hat{C}} = \hat{S}\,\grave{\hat{X}} + \hat{C}_\alpha\grave{\alpha}$$

$$(7.17)\qquad \grave{\hat{S}} = \grave{\hat{X}} + \alpha\grave{\hat{U}} + \grave{\hat{U}}\,\grave{\alpha} \qquad\qquad \grave{r}_o = c\,\grave{\hat{C}} + r\,\grave{\hat{C}}\alpha - D\,\grave{\hat{S}} - \hat{S}\,\grave{D}$$

Finally we obtain

$$(7.18)\quad \begin{cases} \grave{f} = (\hat{C}\,\grave{r}_o - r_o\,\grave{\hat{C}})/r_o^2 & \grave{g} = -\grave{\hat{U}}/\sqrt{\mu} \\ \grave{\dot{f}} = \sqrt{\mu}\,(\hat{S}\,\grave{r}_o - r_o\,\grave{\hat{S}})/rr_o^2, & \grave{\dot{g}} = -\grave{\hat{C}}/r \end{cases}$$

which are used in the perturbation differential equations (7.3) for $\grave{\bar{r}}_o$ and $\grave{\dot{\bar{r}}}_o$.

An alternate formulation for obtaining $\grave{f}$, $\grave{g}$, $\grave{\dot{f}}$, and $\grave{\dot{g}}$ has been presented by Pines [20]. Rather than using a Newton-RAphson method to obtain

$X = \sqrt{\mu}\ (E - E_0)$ from (7.7), this method is used to obtain the difference between the current value of the eccentric anomaly E and the initial value E_0 from

$$(7.19)\qquad F(\Theta) = nt - \Theta + \sin\Theta\left(1 - \frac{r_0}{a}\right) + (\cos\Theta - 1)\left(\frac{\bar{r}_0 \cdot \dot{\bar{r}}_0}{a^2 n}\right) = 0$$

where

$$(7.20)\qquad \begin{cases} \Theta = E - E_0 & \dfrac{1}{a} = \dfrac{2}{r_0} - \dfrac{\dot{r}_0^2}{\mu} \\ \dot{r}_0^2 = \sqrt{\dot{\bar{r}}_0 \cdot \dot{\bar{r}}_0} & n = \sqrt{\mu/a^3} \end{cases}$$

Differentiating (7.19), we obtain

$$(7.21)\qquad F'(\Theta) = -1 + \cos\Theta\left(1 - \frac{r_0}{a}\right) - \sin\Theta\left(\frac{\bar{r}_0 \cdot \dot{\bar{r}}_0}{a^2 n}\right).$$

The Newton-Raphson iteration has the form

$$(7.22)\qquad \Theta_{i+1} = \Theta_i - \frac{F(\Theta_i)}{F'(\Theta_i)}$$

and the iteration is terminated whenever

$$(7.23)\qquad |\Theta_{i+1} - \Theta_i| \leq \varepsilon|\Theta_{i+1} + 1|.$$

For $\varepsilon = 10^{-5}$ and Θ_0 equal to the converged to value of Θ at the previous step, only a few iterations are required for convergence.

Next we obtain

$$(7.24)\qquad \begin{cases} f = (a/r_0)\,(\cos\Theta - 1) + 1 & g = t + (\sin\Theta - \Theta)/n \\ \dot{f} = -\,(a^2 n/r r_0)\,\sin\Theta & \dot{g} = (a/r)\,(\cos\Theta - 1) + 1\,. \end{cases}$$

From these we obtain the perturbation derivatiyes required in (7.3) as

$$(7.25)\quad \begin{cases} f^{\backprime} = (\cos\Theta - 1)\,(a/r_0)^{\backprime} - (a/r_0)\sin\Theta\,\Theta^{\backprime} \\ g^{\backprime} = (\sin\Theta - \Theta)\,(1/n)^{\backprime} + [(\cos\Theta - 1)/n]\,\Theta^{\backprime} \\ \dot{f}^{\backprime} = (-\sin\Theta/r)\,(a^2n/r_0)^{\backprime} - (a^2n/rr_0)\,\cos\Theta\,\Theta^{\backprime} \\ \dot{g}^{\backprime} = [(\cos\Theta - 1)/r]\,a^{\backprime} - (a/r)\sin\Theta\,\Theta^{\backprime} \end{cases}$$

where the perturbation derivatives of the variables a, Θ, n, a/r_0, and a^2n/r_0 with respect to time are

$$(7.26)\quad a^{\backprime} = (2a^2/\mu)\,\dot{\bar{r}}\cdot\dot{\bar{r}}^{\backprime}$$

$$(7.27)\quad (1/n)^{\backprime} = (3a/n\mu)\,\dot{\bar{r}}\cdot\dot{\bar{r}}^{\backprime}$$

$$(7.28)\quad \Theta^{\backprime} = [(1 - \cos\Theta)r_0an]\,\bar{r}\cdot\dot{\bar{r}}^{\backprime} + [(g - 3t + (r/an)\sin\Theta)/r_0an]\,\dot{\bar{r}}\cdot\dot{\bar{r}}^{\backprime}$$

$$(7.29)\quad (a/r_0)^{\backprime} = (ag/r_0{}^3)\,\bar{r}\cdot\dot{\bar{r}}^{\backprime} - (a/r_0{}^3)\Big\{[\bar{r}_0\cdot\dot{\bar{r}}_0/a^2n^2]\,[3\Theta - g - (r/an)\sin\Theta] + (r_0/an)\,[(\bar{r}\cdot\dot{\bar{r}}/a^2n^2)\sin\Theta - (2r/an)\cos\Theta]\Big\}\,\dot{\bar{r}}\cdot\dot{\bar{r}}^{\backprime}$$

$$(7.30)\quad (a^2n/r_0)^{\backprime} = -\dot{\bar{r}}\cdot\dot{\bar{r}}^{\backprime}/r_0n + an\,(a/r_0)^{\backprime}$$

8. METHODS FOR APPROXIMATING DRAG IIPs BETWEEN COMPUTATIONAL CYCLES.

A linear extrapolation of previously computed drag IIPs is currently being used to obtain approximations to drag IIPs between computational cycles. For example, suppose a drag IIP output is required 10 times per second but drag IIPs are computed only 10/n times per second. Let $\bar{R}_{i-n}$ and $\bar{R}_i$ be the computed drag IIPs corresponding to the initial (range) times t_{i-n} and t_i respectively. Then the linear extrapolation $\tilde{R}_{i+k}$ to the drag IIP corresponding to the initial time t_{i+k} $(1 \leq k < n)$ is

(8.1) $\tilde{R}_{i+k} = \bar{R}_i + (\bar{R}_i - \bar{R}_{i-n})k/n,$

and the time to impact $\tilde{T}_{i+k}$ is extrapolated by

(8.2) $\tilde{T}_{i+k} = T_i + (T_i - T_{i-n})k/n,$

where T_i and T_{i-n} are impact times corresponding to initial times t_i and t_{i-n} respectively.

When n is small (say 2) then (8.1) furnishes a good approximation for drag IIPs at intermediate times. However, when n is larger (say 10 or 20), then (8.1) is not as good because it assumes that the drag IIPs will continue in a straight line. Furthermore, it does not account for the new values of the missile's initial position and velocity at these intermediate times. Therefore, the following alternative methods have been investigated.

A linear extrapolation of the ratios of the differences between the drag IIPs from previous computational cycles to the differences between the corresponding Kepler IIPs, in conjunction with the Kepler IIP for the current value of the missile's initial position and velocity, can be used to obtain a much better approximation to the current drag IIP. This method requires about the same amount of computational time as (8.1), yet is approximately six times more accurate, provided that the Kepler IIPs are strictly monotone as a function of initial range times. Furthermore, it does account for new values of the missile's initial position and velocity, since the Kepler IIP for these values is part of the computation in this method. It is described as follows.

Let $\bar{R}_{i-2n}$, $\bar{R}_{i-n}$, and $\bar{R}_i$ denote the computed drag IIPs for initial times t_{i-2n}, t_{i-n}, and t_i, with T_{i-2n}, T_{i-n}, and T_i their corresponding impact times. Similarly let $\bar{r}_{i-2n}$, $\bar{r}_{i-n}$, and $\bar{r}_i$ denote the Kepler IIPs for initial times t_{i-2n}, t_{i-n}, and t_i, with K_{i-2n}, K_{i-n}, and K_i their corresponding Kepler impact times. Let $\bar{P}_i$ denote the ratio of the difference between the most recently computed drag IIPs $\bar{R}_i$ and $\bar{R}_{i-n}$ to the difference between the corresponding Kepler IIPs $\bar{r}_i$ and $\bar{r}_{i-n}$, i.e.

(8.3) $\bar{P}_i = \dfrac{\bar{R}_i - \bar{R}_{i-n}}{\bar{r}_i - \bar{r}_{i-n}}$

Similarly, let $\overline{P}_{i-n}$ denote this ratio for the previously computed set of drag and Kepler IIPs, i.e.

$$(8.4) \qquad \overline{P}_{i-n} = \frac{\overline{R}_{i-n} - \overline{R}_{i-2n}}{\overline{r}_{i-n} - \overline{r}_{i-2n}}$$

A linear extrapolation $\tilde{P}_{i+k}$ to this ratio corresponding to the initial time t_{i+k} $(1 \leq k < n)$ is then given by

$$(8.5) \qquad \tilde{P}_{i+k} = \overline{P}_i + (\overline{P}_i - \overline{P}_{i-n})\ k/n.$$

If $\overline{r}_{i+k}$ denotes the Kepler IIP for initial time t_{i+k} with time to impact K_{i+k}, then the drag IIP for this initial time $\tilde{R}_{i+k}$ is approximated by

$$(8.6) \qquad \tilde{R}_{i+k} = \tilde{P}_{i+k}\ (\overline{r}_{i+k} - \overline{r}_i) + \overline{R}_i\ .$$

Similarly, its time to impact T_{i+k} is approximated by

$$(8.7) \qquad \tilde{T}_{i+k} = Q_{i+k}\ (K_{i+k} - K_i) + T_i\ ,$$

where

$$(8.8) \qquad Q_{i+k} = Q_i + (Q_i - Q_{i-n})\ k/n$$

and

$$(8.9) \qquad Q_i = \frac{T_i - T_{i-n}}{K_i - K_{i-n}}\ ,$$

$$(8.10) \qquad Q_{i-n} = \frac{T_{i-n} - T_{i-2n}}{K_{i-n} - K_{i-2n}}\ .$$

A third method in which a quadratic extrapolation of drag IIPs, considered as a function of Kepler IIPs, instead of range time, can be used to yield a slight improvement over the previous method for about the same amount of computation time.

Using the notation from the previous section, we obtain the approximation $\tilde{R}_{i+k}$ to the drag IIP for initial range time t_{i+k} as

$$
\begin{aligned}
(8.11)\qquad \tilde{R}_{i+k} = \frac{1}{\bar{r}_{i-n} - \bar{r}_{i-2n}} \Bigg\{ & \frac{\bar{r}_{i-n} - \bar{r}_{i+k}}{\bar{r}_i - \bar{r}_{i-2n}} \Big[\bar{R}_{i-2n} (\bar{r}_i - \bar{r}_{i+k}) \\
& -\bar{R}_i (\bar{r}_{i-2n} - \bar{r}_{i+k}) \Big] - \frac{\bar{r}_{i-2n} - \bar{r}_{i+k}}{\bar{r}_i - \bar{r}_{i-n}} \\
& \Big[\bar{R}_{i-n} (\bar{r}_i - \bar{r}_{i-k}) - \bar{R}_i (\bar{r}_{i-n} - \bar{r}_{i+k}) \Big] \Bigg\}
\end{aligned}
$$

where $\bar{R}_{i-2n}$, $\bar{R}_{i-n}$, $\bar{R}_i$ are the drag IIPs and $\bar{r}_{i-2n}$, $\bar{r}_{i-n}$, $\bar{r}_i$ are the Kepler IIPs for initial range times t_{i-2n}, t_{i-n}, t_i, respectively. Since $\bar{r}_{i+k}$ is the Kepler IIP for initial range time t_{i+k}, it is easily seen that the corresponding approximation $\tilde{R}_{i+k}$ is a function of $\bar{r}_{i+k}$. A similar expression is valid for approximating the time to drag impact $\tilde{T}_{i+k}$ as a function of the time to Kepler impact K_{i+k} for initial time t_{i+k}.

In general, higher order extrapolations may yield increasingly worse approximations to the drag IIPs as the order is increased. This would be especially true if the radar-determined position and velocity of the missile were not following a smooth trajectory, in which case an accurate drag IIP would be needed most. In fact, "If polynomial extrapolation must be done with poorly behaved functions, then very low degree extrapolation is usually the safest, but even this should be carried out only for values of x very close to the tabulated region," [28, p. 58].

9. SUMMARY OF RESULTS AND CONCLUSIONS. In general, the method of variation of parameters took about twice as long as Cowell's method for the computation of the same drag IIPs, whereas Encke's method took about four times as long. Originally it was hoped that these two methods would offer an improvement over Cowell's method which is presently being used. However, these two methods would have been an improvement over Cowell's method only if the perturbation accelerations were small compared with the gravity accelerations and the trajectory was not highly eccentric. Since both of these conditions are more likely to exist for artificial satellites, rather than for ballistic missiles, it becomes clear why Encke's method and the method of variation of parameters are preferred for orbit determination of artificial satellites and Cowell's method is preferred for trajectory determination of ballistic missiles. It should be noted, however, that for a relative trunction error of 10^{-5}, all three methods produced drag IIPs which agreed to within a few feet, with the exception of Herrick's version of the method of variation of parameters. An error exists either in the equations themself or else in the coding

of these equations. Since the use of Herrick's method did not result in a reduction in computational time as compared with Cowell's method, an attempt to find this error was not seriously persued.

A comparison of the various numerical integration methods revealed that for a specified relative truncation error, the rational extrapolation method yielded approximately the same drag IIPs as the fourth-order Adams-Moulton (4th A-M) method with slightly less computational time and 1/2 to 2/3 as many functional evaluations required. When a relative truncation error of 0.01 was specified, the drag IIPs differed from those computed with a relative truncation error of 10^{-5} by less than 20 feet with about a three-fold decrease in computer time required for both methods.

Since the truncation error in the fourth-order Runge-Kutta method (4th R-K) cannot be easily determined, the best way of comparing it with the other two methods is to specify the number of steps or iterations, rather than the relative truncation error, for all three methods and compare their drag IIPs and corresponding times to impact with those obtained using a relative truncation error of 10^{-5}. When 40 steps and Beta = 100.0 were specified, the rational extrapolation method required 120 functional evaluations, and yielded drag IIPs 400-500 feet short, and corresponding times to impact were about 1.5 seconds short. By comparison, the 4th A-M method required 90 functional evaluations and yielded drag IIPs 400-1000 feet short with times to impact about 2 seconds short, whereas the 4th R-K method required 160 functional evaluations, yielded drag IIPs 500-2000 feet short with times to impact about 2 seconds short, and required 1-1/4 to 1-1/2 times as much computer time as the 4th A-M.

When 20 steps and Beta = 100.0 were specified all three methods required about the same amount of computer time per drag IIP (60-75 milliseconds). The 4th A-M method yielded drag IIPs 4000-5000 feet short with times to impact about 6 seconds short, the rational extrapolation method yielded drag IIPs 8000-13000 feet short with times to impact about 14 seconds short, and the 4th R-K method yielded drag IIPs 6000-15000 feet short with times to impact about 20 seconds short. When 10 steps were attempted, all three methods failed to converge.

An attempt was made to improve the numerical integration process by coding the Gauss-Jackson (Σ^2) method which, according to Herrick [13, p. 1], is approximately fifty times more effective than the 4th R-K method for moderate-eccentricity orbits. Since Cowell's method had already been seen to be far more effective than either Encke's method or either version of the method of variation of parameters for computing missile trajectories, the Σ^2 method was coded only for Cowell's method. Implementing the code from the discussion in [13] required some effort since no codes or algorithms appeared in [13]. Two versions of the Σ^2 method were coded. In the first version, three forward and three backward values of the components of the initial acceleration vector of the missile were required to start the difference tables. This was the version presented in [13], which is more appropriate for the determination of satellite orbits rather than missile trajectories, since backward values of acceleration are meaningless for a missile at launch time or shortly thereafter. In the second version six forward values and no backward values of the components of the initial acceleration vector were required to start the difference tables. In both versions three methods of estimating these values are available, namely Taylor series approximations, Kepler approximations, and 4th R-K approximations. Of these the 4th R-K approximations provided the most accurate estimates of the forward and backward values of the initial acceleration vector. However, regardless of which of these three methods was used, the iteration process on the difference tables for starting the Σ^2 method diverged, unless the initial step size was choosen to be quite small, usually less than 0.25 seconds. When a step size twice as large was used (0.46921559), the velocities began to ocillate prior to impact, resulting in a computed drag IIP and time to impact considerably shorter than the correct values. No oscillation of velocities was observed for any of the other numerical integration methods regardless of how large a step size was used. However, when a step size of 0.23408923 was used, the computed drag IIPs by the Σ^2 method agreed with those obtained using the 4th A-M method with a relative truncation error of 10^{-5} to within 10 feet, and no oscillation of velocities was observed. Therefore, it appears that the Σ^2 method is unstable for large step sizes and cannot be recommended, unless it can be modified in some way to eliminate these instabilities. The unstable nature of the Σ^2 method was not mentioned by Herrick [13], and in fact may have been unknown to him since he illustrated the Σ^2 method only for a very simple equation of motion, with no drag terms and a circular orbit. His method of starting the difference tables is not sufficiently refined to accurately account for drag, so a modification to his starting procedure is required. On the other hand such a modification may not be sufficient for the Σ^2 method to be comparable to, or to even surpass, the other numerical integration methods. The Σ^2 method as presented in [13] uses up to sixth-order differences, which is equivalent to a seventh-order method of the Adams type. Higher order methods are better suited to smooth problems such as satellites in nearby circular orbits with minimal perturbation forces rather than missiles in highly, eccentric trajectories subjected to high perturbation forces. This fact has been demonstrated in our current real-time program for the cases in which $\beta < 200$ by observing that drag IIPs are more efficiently computed using a 2nd R-K method with 20 steps than a 4th R-K method with 10 steps, although both methods require approximately the same amount of computation time. In fact in the discussion of the variable order, variable step Adams method in [12, p. 127], appears "... we shall see that the lower the order the more stable the method ..." and "Stiff problems will drive the order and the step size down to small values." Therefore, it seems highly unlikely that the Σ^2 method will be an improvement over other numerical methods currently available, at least for missile trajectory problems.

Of the three methods for approximating drag IIPs between computational cycles, the method of quadratic extrapolation of drag IIPs considered as a function of Kepler IIPs yielded the most accurate approximations to the more accurately computed values of drag IIPs using a relative truncation error of 10^{-5}, while requiring about the same amount of computer time (1-2 microseconds) as the other two methods.

10. RECOMMENDATIONS FOR FURTHER STUDY. Based upon these results, the implementation of different numerical integration methods most likely will not provide a significant improvement in the efficiency of computing drag IIPs. A significant improvement will probably be realized only if the equations of motion for missile trajectories are developed in a manner similar to the development of Encke's method

or the method of variation of parameters for satellite orbits. For both of these methods, the reference orbit is obtained from Kepler's equations of motion and is a close approximation to the true orbit of the satellite. However, the reference trajectory obtained from Kepler's equations of motion is not a close approximation to the true drag trajectory of a ballistic missile. For these reasons, Encke's method and the method of variation of parameters are in general an improvement over Cowell's method for computing satellite orbits but are less effective than Cowell's method for computing missile trajectories, due to the much larger perturbation forces experienced by the missile.

Two solutions to this problem are possible. In the first case, the reference trajectory in Encke's method need not be a Kepler trajectory. In fact the previously computed drag trajectory, displaced by an appropriate amount would probably be a better approximation to the drag trajectory than the Kepler trajectory. Other possibilities also exist. A linear extrapolation of two previous drag trajectories or a quadratic extrapolation of three previous drag trajectories may be used to obtain reference trajectories. Encke's method could then be developed for the differences between the true drag trajectory and one of these reference trajectories. In view of the accurate approximations to drag IIPs between computational cycles obtained from the method of quadratic extrapolation of drag IIPs considered as a function of Kepler IIPs, the reference trajectory in this case should be a much closer approximation to the true drag trajectory than the Kepler trajectory was. If this is indeed the case, then the corresponding equations of motion for Encke's method could be numerically integrated using far fewer steps than for Cowell's method, and hopefully result in a considerable savings in time to compute drag IIPs.

In the second case, the f and g functions in the solution of Kepler's equations of motion (6.17) or the method of variation of parameters (7.1-7.2) can also be developed as a time series [29, pp. 107-111], known as the f and g series. Usually such a Taylor series type of solution to an initial value problem is "generally impractical from a computational point of view", [30, p. 365]. In fact "... the necessity of calculating the higher derivatives makes Taylor's algorithm completely unsuitable on high-speed computers for general integration purposes", [31, p. 330]. However, during a recent conversation [32], this author was made aware of some recent results using the f and g series for missile trajectory prediction, including the effects of drag and the earth's oblateness. According to one of the reports [33, p. 19], "The f and g series procedure provides equivalent or greater accuracies than numerical integration of missile trajectory prediction and reduces computer run time by a factor of from seven to ten due to the requirement for less iterative steps for a given impact calculation." A second report uses the f and g series technique developed in [33] to determine and display to a Missile Flight Control Officer (MFCO) "The geographical distribution of debris impact coordinates that would result if the missile were destroyed," [34, p. 287]. According to this report [34, p. 288], the method presented "... uses the actual vehicle state vector to predict accurately the impact point of a debris particle within milliseconds where previous standard numerical-integration methods ... required over a second." Unfortunately this method has only recently been brought to this author's attention, so that a thorough analysis of it, and subsequent coding for test cases, has not yet been completed.

The method of approximating drag IIPs between computational cycles using a quadratic extrapolation of the three previously computed drag IIPs considered as a function of Kepler IIPs is a considerable improvement over the linear extrapolation method presently being used, provided consecutive pairs of Kepler IIPs are not "too close" together. Since this author has recently been made aware of this condition occuring during the flights of missiles different from those considered in this report, alternative methods for approximating drag IIPs between computational cycles based upon the initial position and velocity of the missile, rather than its Kepler IIP, are being investigated.

11. REFERENCES.

1. Berry, F.N. and W.W. Page, "Athena Impact Prediction for Range Flight Safety," Army Missile Test and Evaluation Directorate, Electro-Mechanical Laboratories Division, Technical Note #103, June 1964.

2. Baker, C.M., "IMPCTR - An Automated Impact Prediction Procedure," Analysis and Computation Directorate, Systems Programming Division, Internal Memorandum NR. 79, April 1968.

3. Falke, J.F., "RTMIN Drag Corrected IIP," NR-AM-R, April 1974 (Rev. June 1975).

4. Falke, J.F., "RTMIN Drag IIP Control Logic," NR-AM-R, August 1977.

5. Berry, F.N., "A Brief Survey of the Differences Between Kepler and Integrated Impact Predictions," Computer Directorate, Computer Systems Divison, Internal Memorandum NR. 9, November 1965.

6. Falke, J.F., "Development of Missile Instantaneous Impact Prediction Schemes for Digital and Hybird Computers," Project Report for CICE 699, NR-AM-R, April 1975.

7. Gear, C.W., "Algorithon 407 - DIFSUB for Solution of Ordinary Differential Equations [D2]," Communication of the ACM, Vol. 14, No. 3, March 1971, pp. 185-190.

8. Gear, C.W., "The Automatic Integration of Ordinary Differential Equations," Communication of the ACM, Vol. 14, No. 3, March 1971, pp. 176-179.

9. Gear, C.W., Numerical Initial Value Problems in Ordinary Differential Equations, Prentice-Hall, Englewood Cliffs, N.J., 1971.

10. Bulirsch, R. and J. Stoer, "Numerical Treatment of Ordinary Differential Equations by Extrapolation," Numer. Math., Vol. 8, 1966, pp. 1-13.

11. Fox, P.A., "DESUB: Integration of A First-Order System of Ordinary Differential Equations," Mathematical Software, J. Rice, ed., Academic Press, New York, 1971, Chapter 9.

12. Shampine, L.F. and M.K. Gordon, Computer Solution of Ordinary Differential Equations: The Initial Value Problem, W.H. Freeman, San Francisco, 1975.

13. Herrick, S., Astrodynamics: Orbit Correction, Perturbation Theory, Integration, Vol. 2, Van Nostrand Reinhold Co., London, 1972.

14. Fox, P., "A Comparative Study of Computer Programs for Integrating Differential Equations," Communication of the ACM, Vol. 15, No. 11, November 1972, pp. 941-948.

15. Hull, T.E., W.H. Enright, B.M. Fellen and A.E. Sedgwick, "Comparing Numerical Methods for Ordinary Differential Equations," SIAM J. Numer Anal., Vol. 9, No. 4, December 1972, pp. 603-637.

16. Shampine, L.F., H.A. Watts, and S.M. Davenport, "Solving Nonstiff Ordinary Differential Equations - The State of the Art,", SIAM Review, Vol. 18, No. 3, July 1976, pp. 376-411.

17. Krogh, F.T., "VODQ/SVDQ/DVDQ - Variable Order Integrators for the Numerical Solution of Ordinary Differential Equations," Section 314 subroutine write-up, Jet Propulsion Laboratory, Pasedena, Calif., 1969.

18. Kuzanek, J.F., "Trip Report - Society for Industrial and Applied Mathematics (SIAM) 1977 Fall Meeting," NR-AR, 4 Nov 77.

19. Deutsch, R., Orbital Dynamics of Space Vehicles, Prentice - Hall, Inc., Englewood Cliffs, 1963.

20. Pines, S., "Variation of Parameters for Elliptic and Near Circular Orbits," The Astronomical Journal, Vol. 66, No. 1, February 1961, pp. 5-7.

21. Kyner, W.T. and M.M. Bennett, "A Modified Encke Special Perturbation Method," The Astronomical Journal, Vol. 71, No. 7, September 1966, pp. 579-584.

22. Stumpff, K. and E.H. Weiss, "A Fast Method of Orbit Computation," NASA Tech Note TN D-4470, April 1968.

23. Duncan, R.C., Dynamics of Atmospheric Entry, McGraw-Hill, New York, 1962.

24. Herrick, S., Astrodynamics, Orbit Determination, Space Navigation, Celestial Mechanics, Vol. 1, Van Nostrand Reinhold, London, 1971.

25. Ralston, A., A First Course in Numerical Analysis, McGraw-Hill, Inc., New York, 1965.

26. Clark, N.W., ANL D 250-DIFSUB (3600 Fortran Program), Argonne National Laboratories, 1966.

27. Clark, N.W., ANL D251-DIFSUB (3600 Fortran Program for use with ANL D250), Argonne National Laboratories, 1966.

28. Hornbeck, R.W., Numerical Methods, Quantum Publishers, Inc., New York, 1975.

29. Escobal, P.R., Methods of Orbit Determination, John Wiley & Sons, Inc., New York, 1965.

30. Dorn, W.S. and D.D. McCracken, Numerical Methods with Fortran IV Case Studies, John Wiley & Sons, Inc., New York, 1972.

31. Conte, S.D. and C. de Boor, Elementary Numerical Analysis: An Algorithmic Approach, 2nd ed., McGraw-Hill, New York, 1972.

32. Olson, D., Pacific Missile Test Center, private communication.

33. Baker, R.M.L., Jr. and N.H. Jacoby, Jr., "A Time Series (F&G Series) Approach to Missile Trajectory Prediction," J. of the Astronautical Sciences, Vol. XXIII, No. 1, January-March 1975, pp. 19-59.

34. Baker, R.M.L., Jr., T.J. Mucha, D.R. Darby, N.H. Jacoby, Jr., A.W. Johnson, and R.E. Ryan, "Range-Safety Debris-Pattern Analysis," J. of the Astronautical Sciences, Vol. XXIII, No. 4, October-December 1975, pp. 287-323.

35. Kuzanek, J.F., "An Investigation of Various Methods to Improve Drag Corrected Instantaneous Impact Predictions for Range Flight Safety," National Range Operations Directorate, Analysis and Computation Division, Internal Memorandum No. 160, January 1979.

SURFACE MISS DISTANCE PROGRAM

Dale McLaughlin
Mathematical Services Branch
Analysis and Computation Division
National Range Operations Directorate
White Sands Missile Range, New Mexico 88002

ABSTRACT. A data product which is frequently required at White Sands Missile Range is miss distance: the position of a specified point on a missile relative to a specified point on the aircraft at which the missile is fired for times close to intercept.

Given this position data of missile relative to target and the attitude of the target, and having a computer model of the target surface, it is possible to find the point on the target surface which is closest to the missile as well as the distance to that point. This information is referred to as surface miss distance.

A program is described which computes surface miss distance and produces plots of the target model and missile trajectory from specified points of view.

1. INTRODUCTION.

Miss distance data is derived from tracking telescope measurements in the following way:

(1) The position of the target is obtained from cinetheodolite or radar.

(2) The vertical and horizontal displacements in the film plane of a point on the missile relative to a point on the target are read from tracking telescope film frames containing both the missile and target (at least two telescopes are required).

(3) The azimuth and elevation of the missile relative to the telescope are computed using the position of the target and displacement of missile from target for each film frame.

(4) The position of the missile is computed using the telescope angles via triangulation.

(5) The missile and target positions are differenced to obtain the relative data which is then smoothed.

(6) The data is interpolated or extrapolated to obtain a predicted minimum miss distance and time of minimum approach.

The attitude (pitch, yaw, roll) of the aircraft is usually obtained from on-board telemetry but in some cases is available from optics.

2. THE MODEL. To facilitate computation of surface miss distance the target surface is broken down into triangular sections which are approximated by plane

triangles. Inputing the model is accomplished by reading in the cartesian coordinates of points on the target surface and the order in which the points are to be connected in order to form the triangles.

One target which is used at WSMR is the PQM102 drone. Our model of the PQM102 consists of detailed measurements made directly from the aircraft (Figure 1). Unfortunately, this is the only target for which such detailed measurements are available. All other models have been created by making measurements on rough drawings such as Figure 2, the PQF86 drone. Measuring the coordinates of the points can be a difficult problem if detailed drawings of the target are not available. To aid in defining the model a program has been written which generates fuselage-like and wing-like surfaces given certain measurements. The fuselage option allows one to read in the coordinates of surface points in a vertical slice of the fuselage and generate a cylinder whose cross-section is shaped like the slice but smaller or larger in radius at specified points along the cylinder (Figure 3). The wing option generates a plane four sided surface of triangles given the coordinates of the four corners of the wing (Figure 4). Sections of the target which do not fall into the category of wing or fuselage must be measured and input point by point (Figure 5). The program produces a fairly good representation of the target depending, of course, on the quality of the drawings and measurements. Figure 6 is the completed PQF86 model made from the drawings in Figure 2.

3. FINDING THE SURFACE MISS DISTANCE. Suppose we are given the components of the missile position $\overline{P}$ w.r.t. a coordinate system B fixed to the target. Let the vertices of a triangle of the target surface model also be given in system B, and denoted as $\overline{X}_1$, $\overline{X}_2$, $\overline{X}_3$. (Figure 7)

Let $\overline{V}_2$ and $\overline{V}_3$ be the vectors from vertex $\overline{X}_1$ to vertex $\overline{X}_2$ and from vertex $\overline{X}_1$ to vertex $\overline{X}_3$, respectively. (Figure 8)

$$\overline{V}_2 = \overline{X}_2 - \overline{X}_1$$

$$\overline{V}_3 = \overline{X}_3 - \overline{X}_1$$

Define an orthonormal basis [$\overline{s}_1$, $\overline{s}_2$] in the plane of the triangle with

$$\overline{s}_1 = \overline{V}_2/||\overline{V}_2|| = (s_{11}, s_{21}, s_{31})$$

$$\overline{s}_2 = \frac{\overline{V}_3 - (\overline{V}_3 \cdot \overline{s}_1)\overline{s}_1}{||\overline{V}_3 - (\overline{V}_3 \cdot \overline{s}_1)||} = (s_{12}, s_{22}, s_{32})$$

Let $\overline{q} = \overline{p} - \overline{X}_1$ and project $\overline{q}$ onto the subspace S spanned by [$\overline{s}_1$, $\overline{s}_2$].

Denote the projection expressed in the B frame by $\overline{W}$. Then $\overline{W}^{(S)}$, $\overline{W}$ expressed in the S frame is

$$\overline{W}^{(S)} = \begin{pmatrix} \overline{q} \cdot \overline{s}_1 \\ \overline{q} \cdot \overline{s}_2 \end{pmatrix} = \begin{pmatrix} W_1 \\ W_2 \end{pmatrix}$$

If the end of $\overline{W}$ is interior to the triangle, then the minimum distance from $\overline{p}$ to the triangle is the length of the orthogonal complement $\overline{n} = \overline{q} - \overline{W}$ where $\overline{W} = M_{b/s}\overline{W}^{(S)}$ and $M_{b/s}$ is the matrix which transforms the S system to the B system.

$$M_{b/s} = \begin{bmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \\ s_{31} & s_{32} \end{bmatrix}$$

In order for the end of $\overline{W}$ to be interior to the triangle the end of $\overline{W}$ must lie between $\overline{V}_2$ and $\overline{V}_3$. (Figure 9)

Expressed in the S system $\overline{V}_2$ and $\overline{V}_3$ are

$$\begin{bmatrix} \overline{V}_2^{(S)} \\ \overline{V}_3^{(S)} \end{bmatrix} = \begin{bmatrix} 1/||\overline{V}_2|| & 0 \\ \dfrac{-(\overline{V}_3 \cdot \overline{s}_1)}{||\overline{V}_2||} & \dfrac{1}{||\overline{V}_3 - (\overline{V}_3 \cdot \overline{s}_1)\overline{s}_1||} \end{bmatrix}^{-1} \begin{bmatrix} \overline{s}_2 \\ \overline{s}_2 \end{bmatrix} =$$

$$\begin{bmatrix} ||\overline{V}_2|| & 0 \\ \overline{V}_3 \cdot \overline{s}_1 & ||\overline{V}_3 - (\overline{V}_3 \cdot \overline{s}_1)\overline{s}_1|| \end{bmatrix} \begin{bmatrix} \overline{s}_1 \\ \overline{s}_2 \end{bmatrix}$$

Thus $\overline{W}$ is between $\overline{V}_2$ and $\overline{V}_3$ if these two conditions hold:

$$W_1 > 0$$

and

$$W_2 < \frac{||\overline{V}_3 - (\overline{V}_3 \cdot \overline{s}_1)\overline{s}_1||}{(\overline{V}_3 \cdot \overline{s}_1)} W_1 .$$

Besides being between $\overline{V}_2$ and $\overline{V}_3$, the end of $\overline{W}$ must be on the same side of the line connecting $\overline{V}_2$ and $\overline{V}_3$ as the origin of system S. Mathematically, this is represented by the inequality

$$W_1 < ||\overline{V}_2|| + \frac{[(\overline{V}_3 \cdot \overline{s}_1) - ||\overline{V}_2||]}{||\overline{V}_3 - (\overline{V}_3 \cdot \overline{s}_1)\overline{s}_1||} W_2 .$$

If the above inequalities are satisfied, the point on the triangle closest to the missile is $\overline{W} + \overline{X}_1$ relative to the B origin. If one of the inequalities is not satisfied the closest point must be along an edge of the triangle. To find the closest point we find the closest point on each of the three edges and then take the minimum of those three distances.

To find the minimum distance to $\overline{V}_2$ (edge 1), define τ_1 by

$$\tau_1 = \begin{cases} 0 & (\overline{V}_2 \cdot \overline{q}) < 0 \\ ||\overline{V}_2|| & (\overline{V}_2 \cdot \overline{q}) > ||\overline{V}_2|| \\ (\overline{V}_2 \cdot \overline{q} & \text{otherwise} \end{cases}$$

The closest point on edge $\overline{V}_2$ is then $\overline{r}_1 = \overline{X}_1 + \tau_1\overline{s}_1$, and the minimum distance to edge 1 is $||\overline{p} - \overline{r}_1|| \equiv d_1$. (Figure 10)

Similarly, we find the minimum distance to $\overline{V}_3$ (edge 2) by

$$\tau_2 = \begin{cases} 0 & (\overline{V}_3 \cdot \overline{q}) < 0 \\ ||\overline{V}_3|| & (\overline{V}_3 \cdot \overline{q}) > ||\overline{V}_3|| \\ (\overline{V}_3 \cdot \overline{q}) & \text{otherwise} \end{cases}$$

$$\overline{r}_2 = \overline{X}_1 + \tau_2\overline{V}_3/||\overline{V}_3||$$

and $$d_2 \equiv ||\overline{p} - \overline{r}_2||$$

and the distance to $(\overline{V}_3 - \overline{V}_2)$ (edge 3) by

$$\tau_3 = \begin{cases} 0 & (\overline{V}_3 - \overline{V}_2) \cdot \overline{q} < 0 \\ ||\overline{V}_3 - \overline{V}_2|| & (\overline{V}_3 - \overline{V}_2) \cdot \overline{q} > ||\overline{V}_3 - \overline{V}_2|| \\ (\overline{V}_3 - \overline{V}_2) \cdot \overline{q} & \text{otherwise} \end{cases}$$

$$\overline{r}_3 = \overline{X}_2 - \tau_3(\overline{V}_3 - \overline{V}_2) / ||\overline{V}_3 - \overline{V}_2||$$

and $$d_3 = ||\overline{p} - \overline{r}_3|| .$$

The minimum distance to the triangle is then

$d = \min(d_1, d_2, d_3)$ and the closest surface point is

$\overline{r} = \overline{r}_i$ such that $d_i = d$.

In order to cut down on computation, the distances to the vertices of each triangle are computed, and only the K triangles with the K closest vertices are considered. This could lead to a less than optimal solution. How much less than optimal depends on the size and arragnement of the triangles. In tests using the PQF102 model, there has been very little difference in distance between the five closest triangles. To be optimal, all triangles j whose closest vertex is closer than Z_j + D min should be considered. Here Z_j is the maximum distance between any point interior to triangle j and any vertex of triangle j, and D min is the minimum distance of the missile to any triangle vertex.

After the closest surface point has been determined, a brief description of the area in which the surface point is contained is printed as well as the $\overline{p}$, $\overline{r}$, and d.

4. REMOVING HIDDEN LINES AND PLOTTING. The program offers two options for plotting:

(1) Plot specified views of the target model and the missile trajectory. (Figure 11)

(2) Plot specified views of the target model plus a model of the missile in its correct position and attitude at some specified time. (Figure 12)

The views in the above options may be specified relative to either the target fixed frame or a ground fixed frame.

In order to make the plots nicer to look at and easier to interpret, it is desirable to remove hidden lines (i.e., lines on surfaces hidden by other surfaces). Many hidden lines can be removed by the following procedure:

Given a triangle of the target model, compute the normal to the triangle surface which points outward from the plane. If the outward normal is pointing away from the viewer, do not plot the triangle.

This algorithm removes lines on surfaces on the "back" of the target relative to the viewer, but leaves such surfaces as parts of wings that are hidden by the fuselage. Modifications are planned to remove these remaining hidden lines.

5. ACKNOWLEDGEMENTS. The algorithm for computing the minimum distance to a triangle was suggested by Mr. William S. Agee of WSMR.

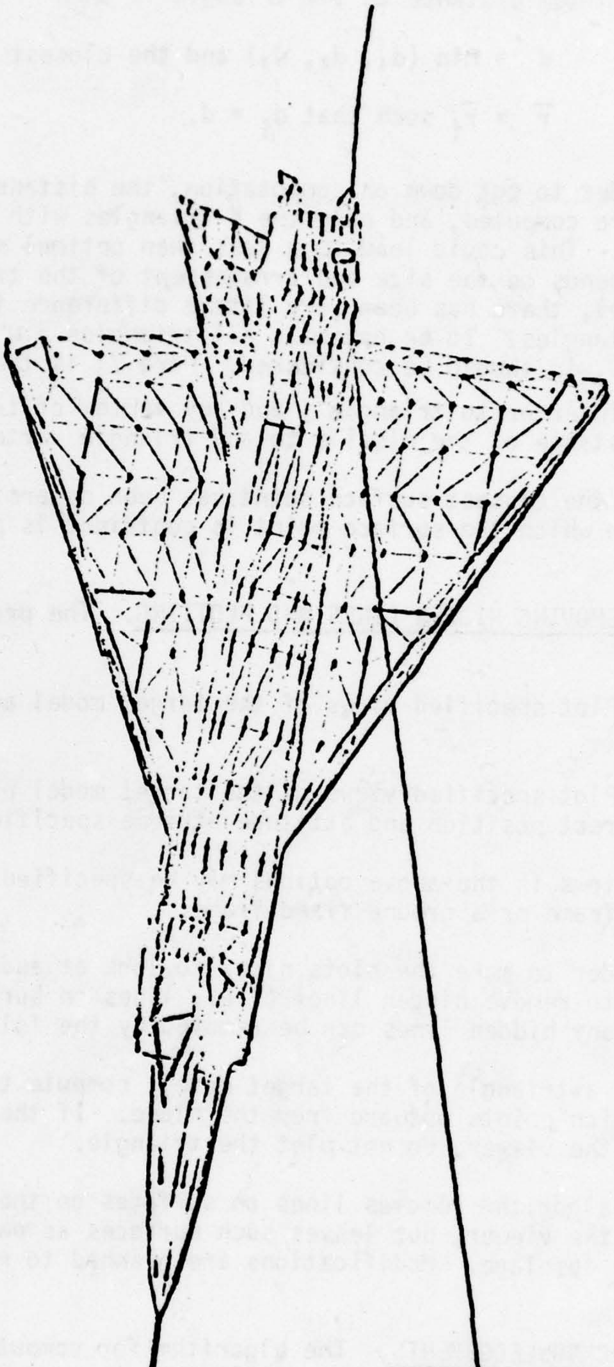


Figure 1

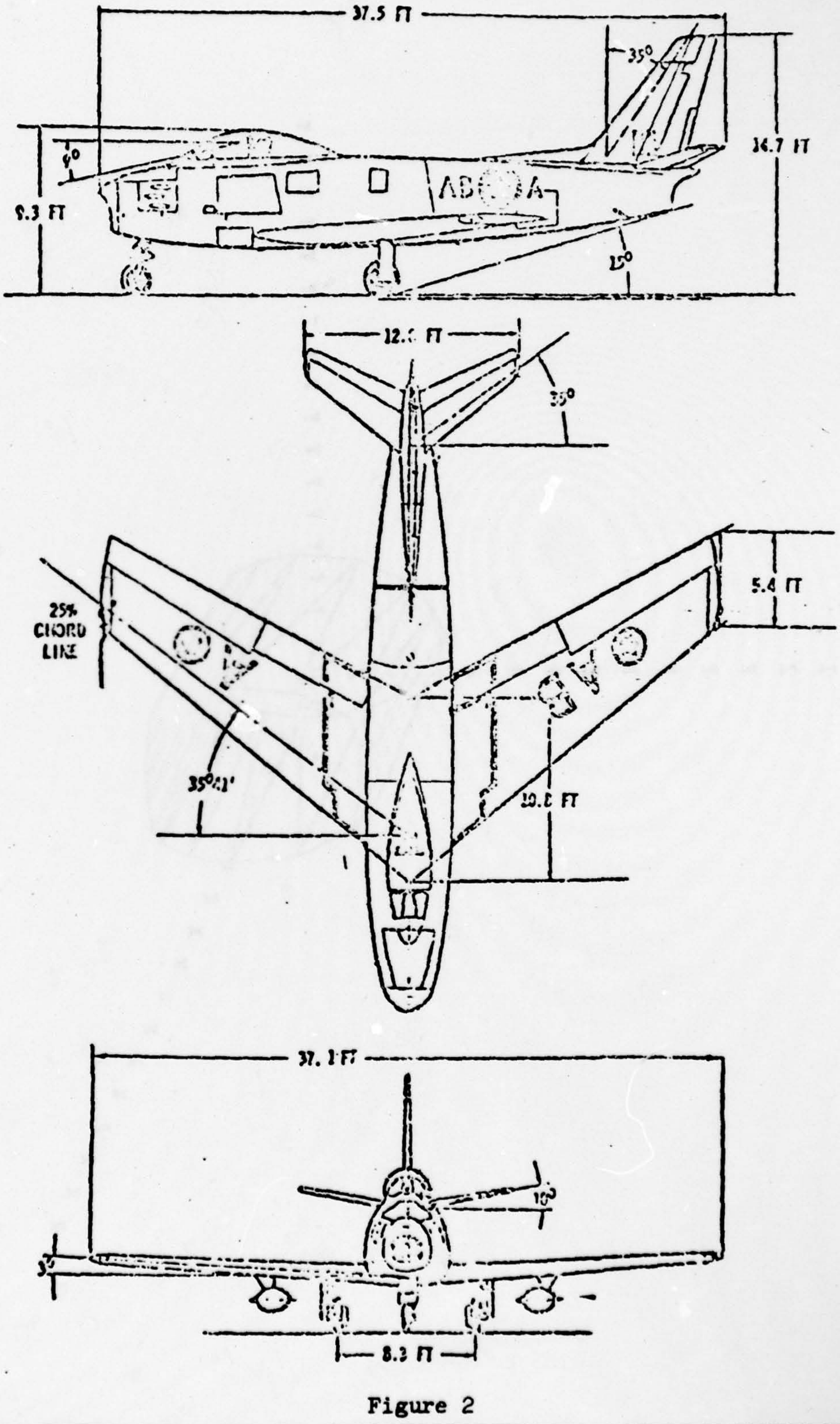

37.5 FT
35°
14.7 FT
6°
9.3 FT
25°
12.8 FT
35°
5.4 FT
25% CHORD LINE
35°41'
10.8 FT
37.1 FT
10°
3°
8.3 FT

Figure 2

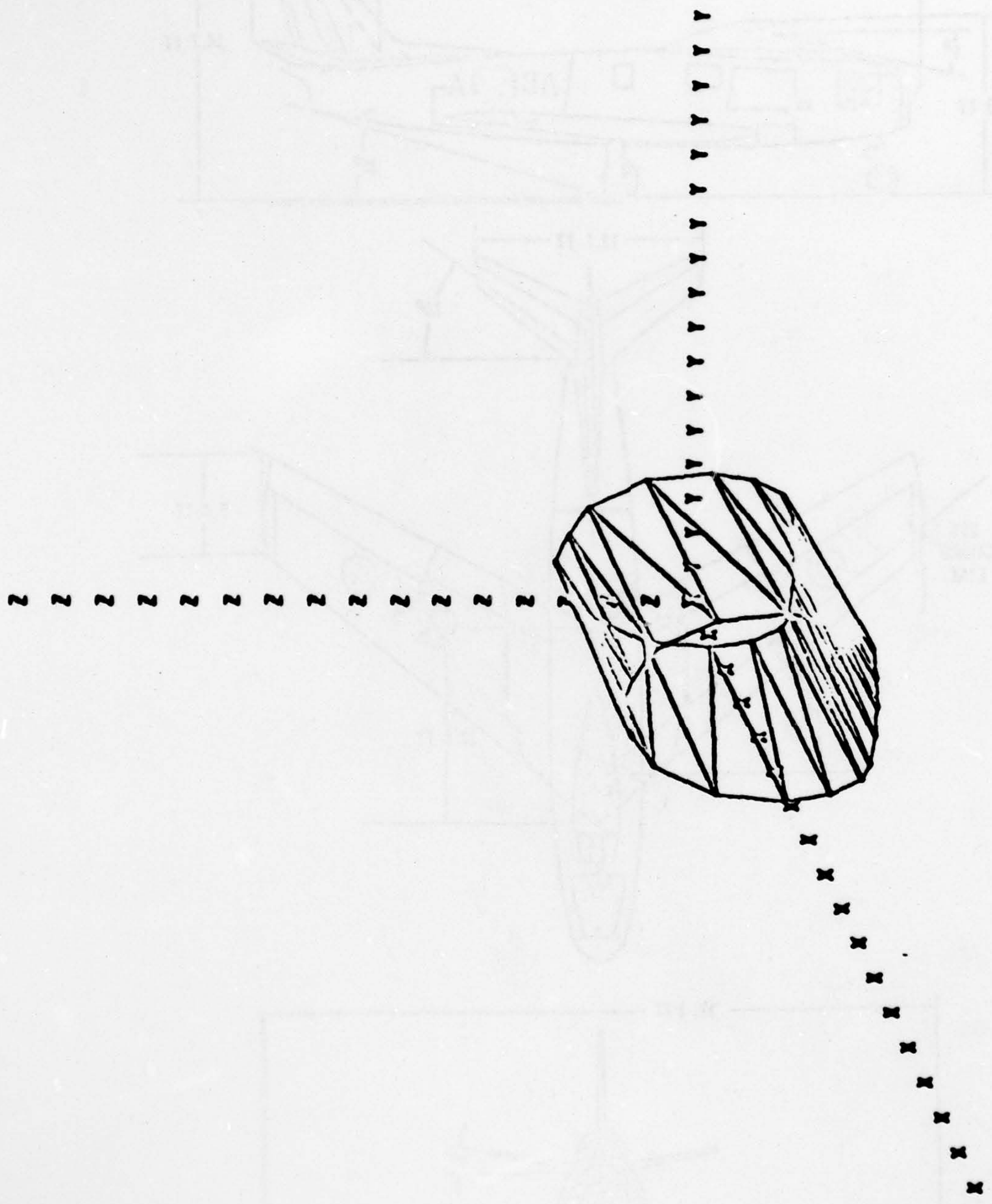


Figure 3

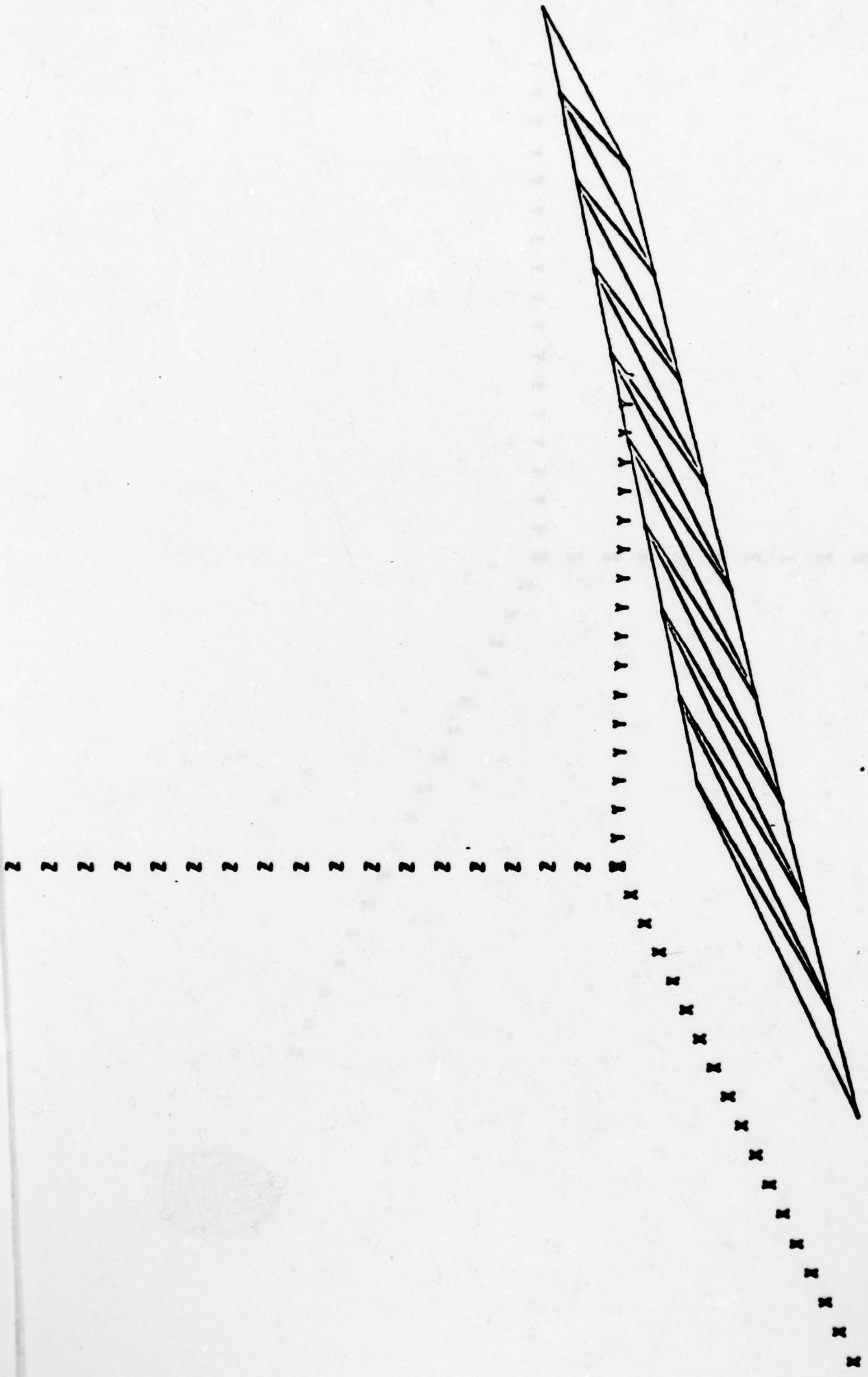
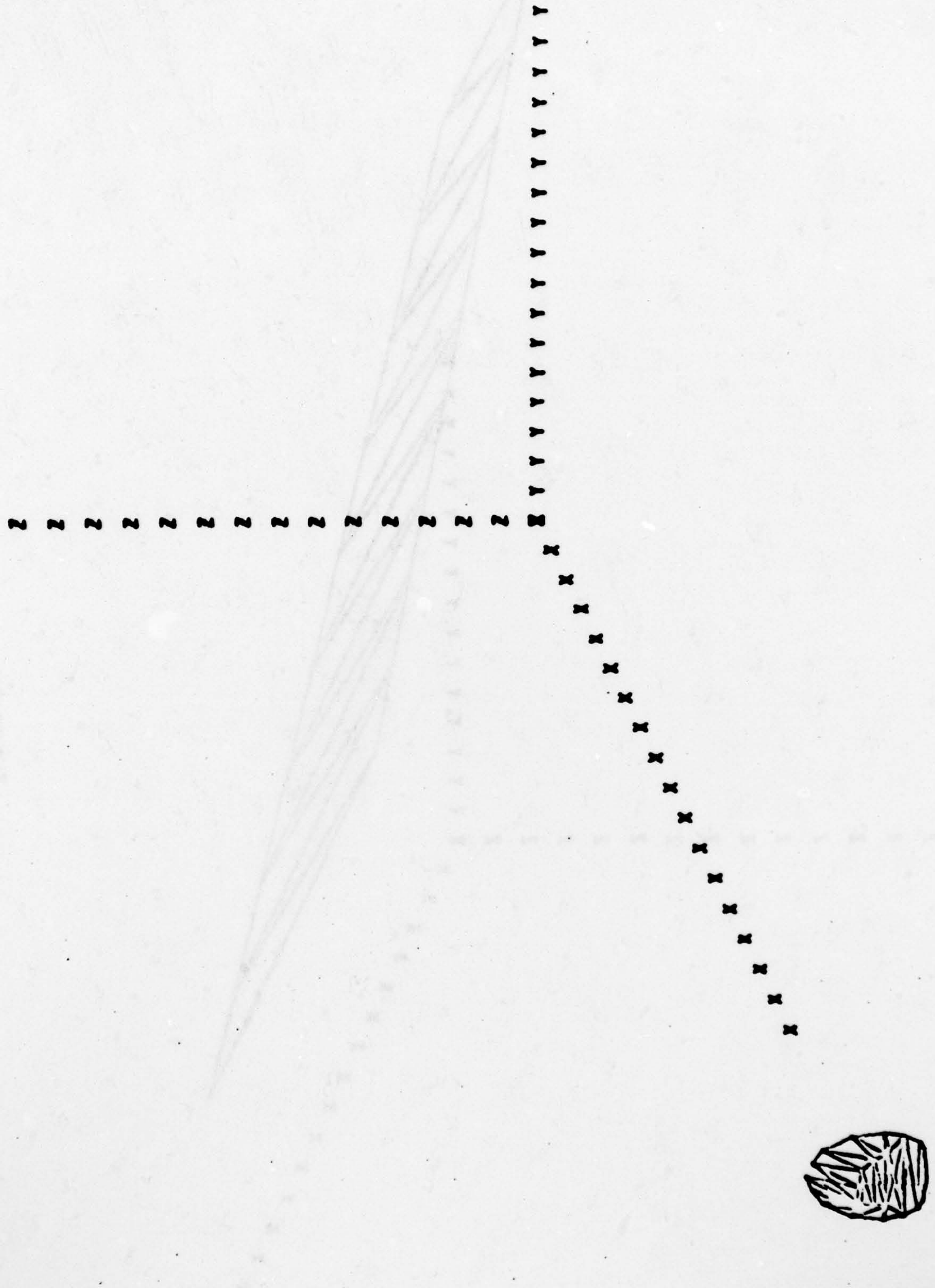


Figure 4



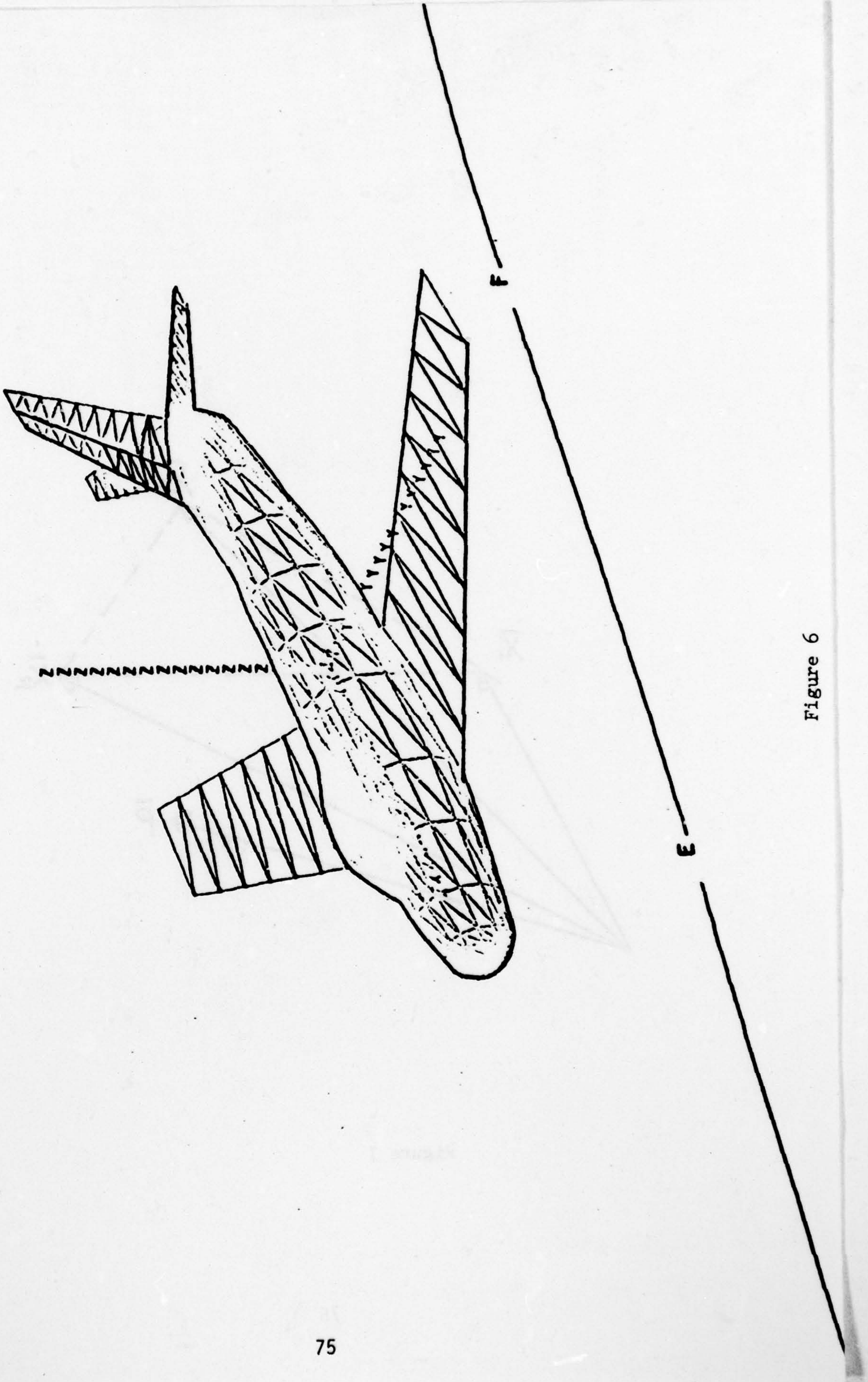


Figure 6

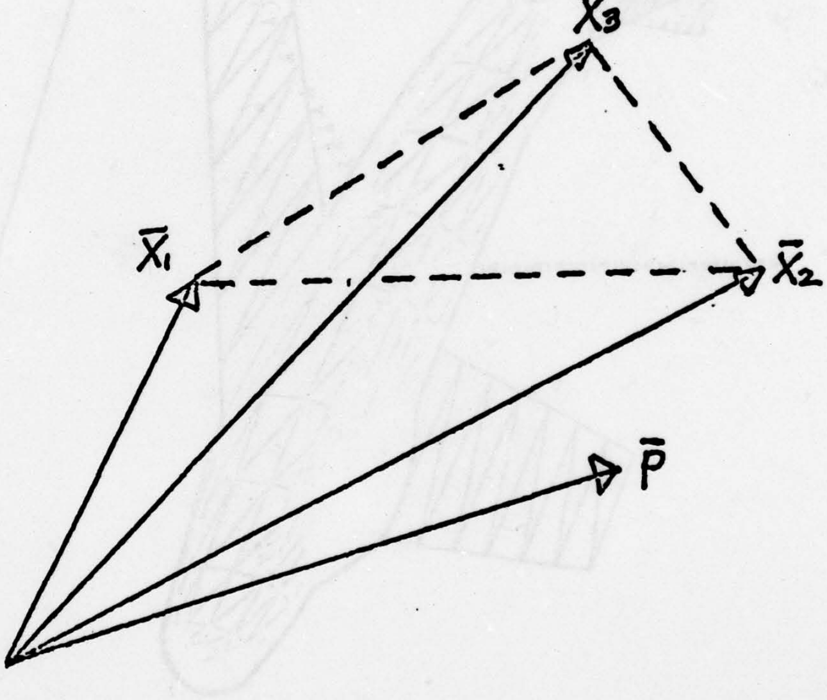


Figure 7

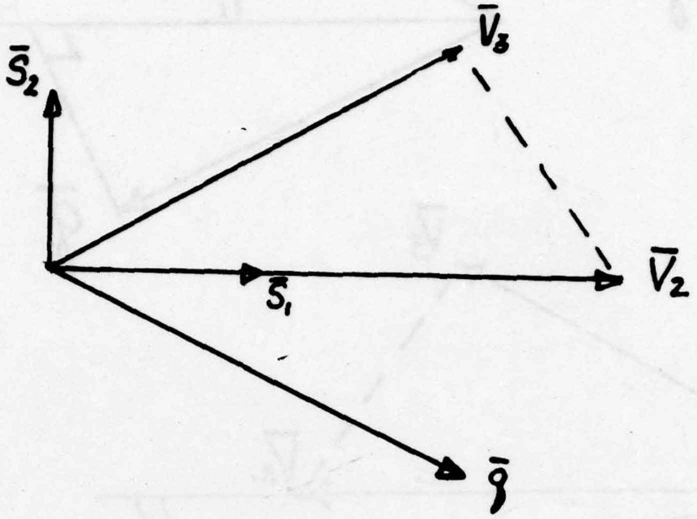


Figure 8

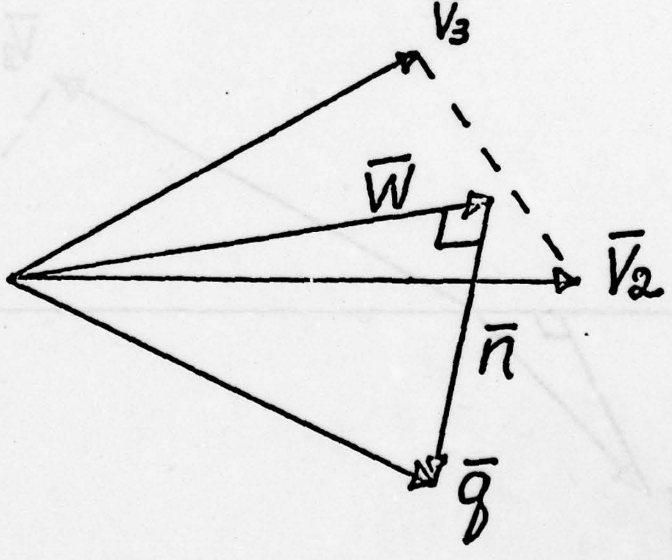
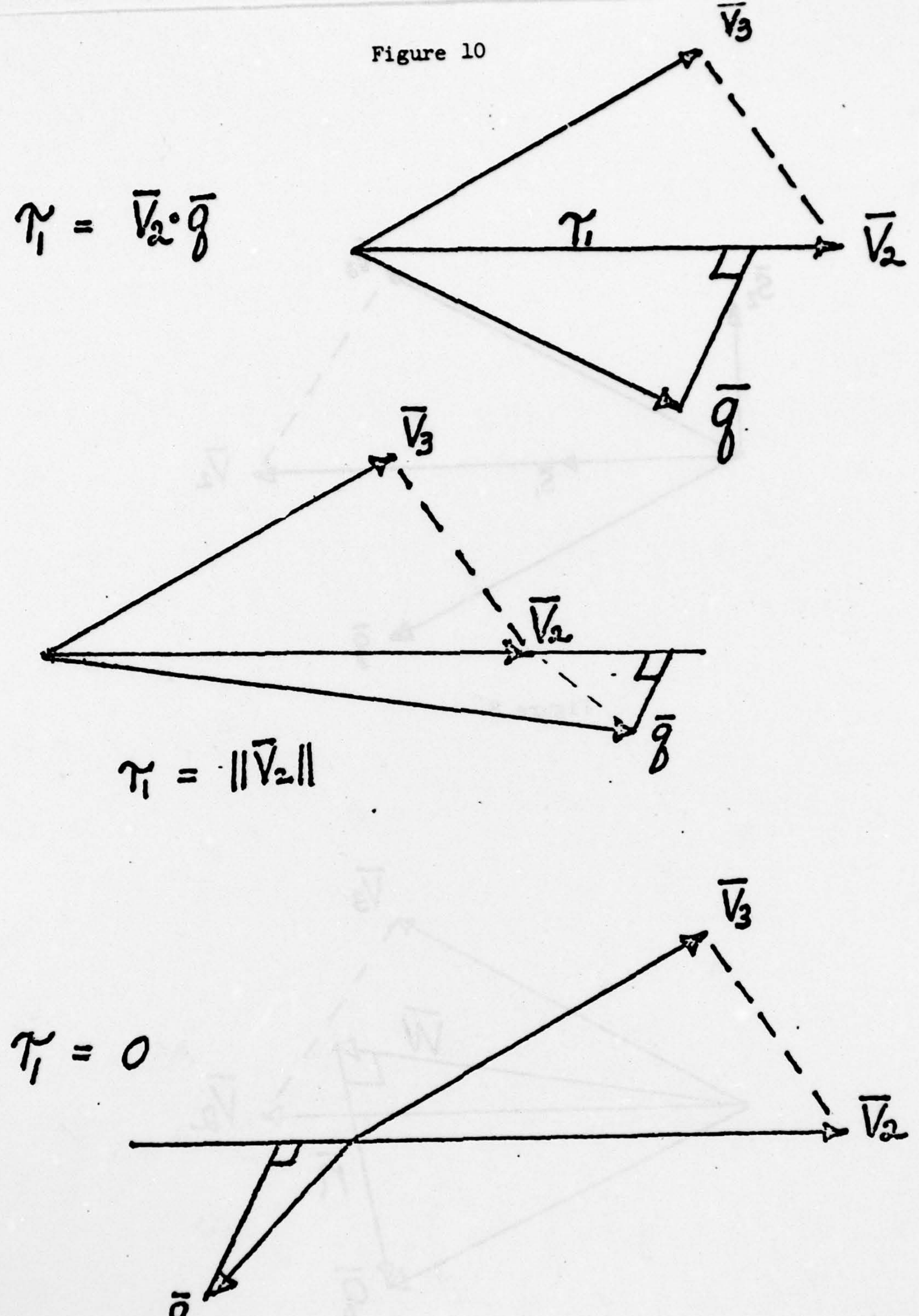


Figure 9

Figure 10



$$\tau_1 = \begin{cases} 0 & (\bar{V}_2 \cdot \bar{q}) < 0 \\ \|\bar{V}_2\| & (\bar{V}_2 \cdot \bar{q}) > \|\bar{V}_2\| \\ (\bar{V}_2 \cdot \bar{q}) & \text{otherwise} \end{cases}$$

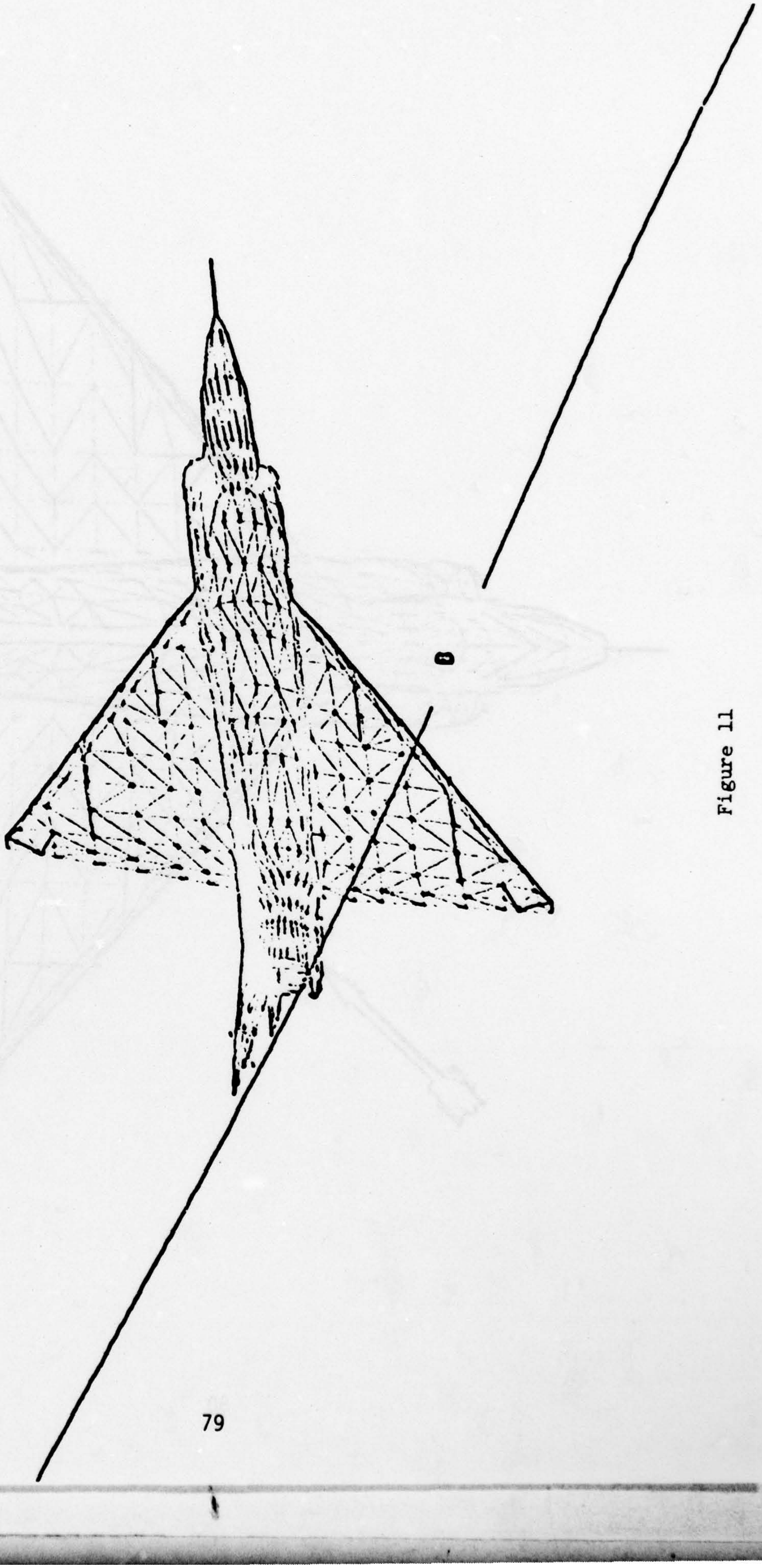


Figure 11

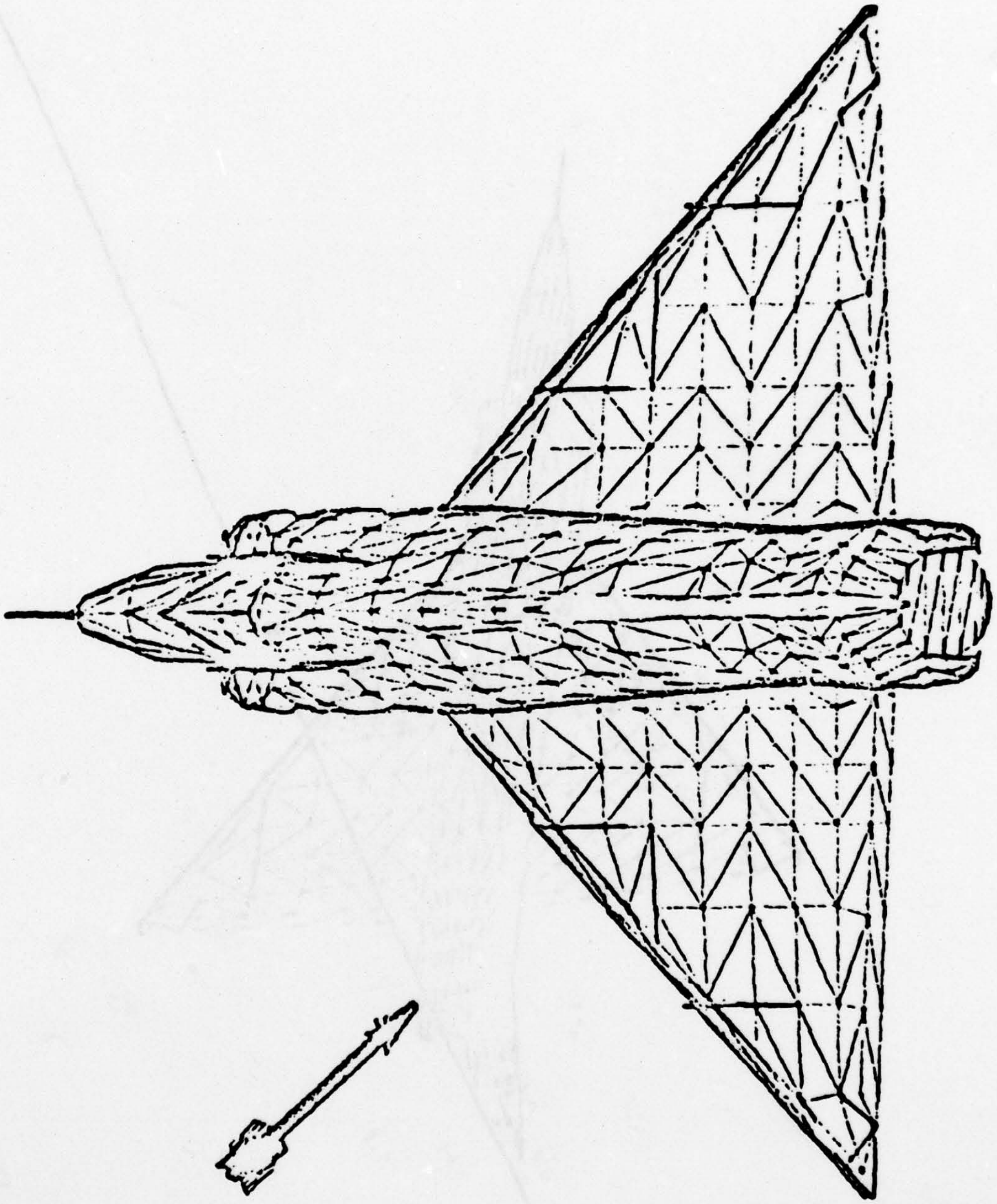


Figure 12

THE OTT FUNCTIONS FOR NAVAL GUNFIRE CONTROL SYSTEMS

Garland H. Ott
Naval Surface Weapons Center
Dahlgren Laboratory
Dahlgren, Virginia 22448

ABSTRACT

Since 1964, when work began on the first Naval Digital Gunfire Control System (GFCS), the Naval Surface Weapons Center (NAVSWC) has conducted extensive studies on the generation of ballistic data needed for computation of gun orders in fire control systems. These studies led to the development of the Ott Functions currently being used in the MARK 86 GFCS. This system which controls one or more Five-Inch 54-Caliber guns is the Navy's most widely used digital GFCS. These Ott Functions represent a giant step forward from earlier ballistic functions in accuracy, storage and computation time.

With the imminent introduction of the Five-Inch Guided Projectile into the Navy's arsenal, the Ott Functions have been modified to generate the ballistic data needed in the use of the Guided Projectile with the MARK 86 GFCS. This paper discusses the Guided-Projectile modifications to the Ott Functions and the resulting impact on accuracy, storage and computation time.

PARTICLE EQUATIONS OF MOTION

A. THREE FORCES CONSIDERED

1. GRAVITY (ALTITUDE)
2. DRAG (MACH NUMBER)
3. THRUST* (TIME)

B. TWO-DIMENSIONAL

1. DOWN-RANGE AND ALTITUDE
2. DRIFT IS SEPARATE INTEGRAL

C. FORM FACTORS

1. DRAG
2. THRUST
3. DRIFT

BALLISTIC VARIABLES

1. SLANT RANGE TO AIM POINT
2. BEARING OF AIM POINT
3. ELEVATION OF AIM POINT
4. INITIAL VELOCITY
5. BALLISTIC DENSITY
6. BALLISTIC TEMPERATURE
7. MOTOR TEMPERATURE *
8. WINDS
9. PROJECTILE WEIGHT
10. EQUIVALENT DENSITY CORRECTION
11. OWN SHIP MOTION

BALLISTIC COMPUTATION METHODS

A. TABLE LOOKUP

B. BALLISTIC FUNCTIONS

1. MARK 68 ANALOG
2. LOCKHEED FUNCTIONS
3. SIGNAAL FUNCTIONS
4. OTT FUNCTIONS

C. NUMERICAL INTEGRATION

1. RUNGE-KUTTA SECOND ORDER
2. RUNGE-KUTTA FOURTH ORDER
3. OBRECHKOFF

RANGE OF CONDITIONS FOR FUNCTIONALIZATION

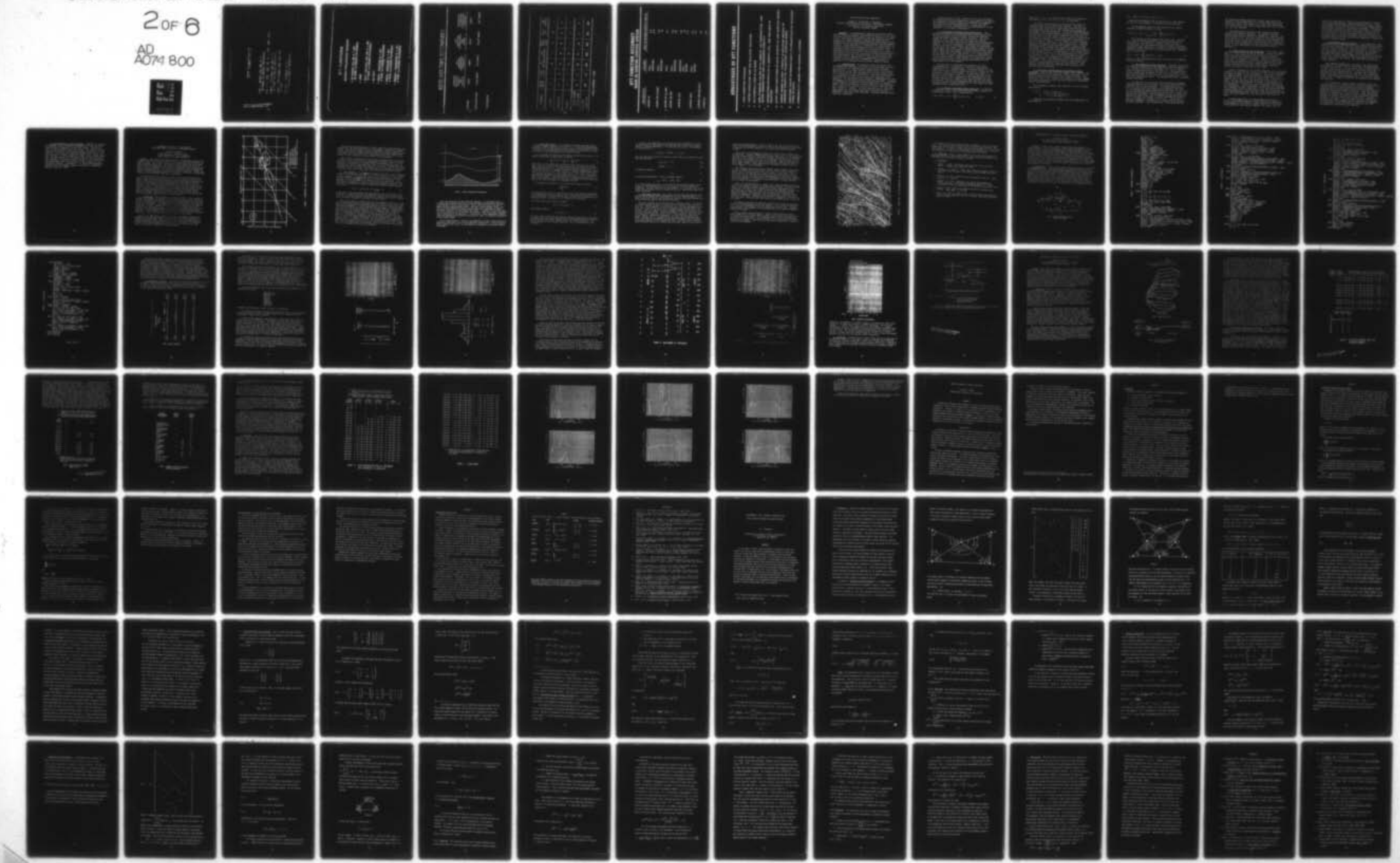
VARIABLE	TYPE OF DISTRIBUTION	MINIMUM/MAXIMUM	MEAN	STANDARD DEVIATION
ELEVATION ANGLE	UNIFORM	0°/65°		
INITIAL VELOCITY	TRUNCATED NORMAL	2660/2960 FPS	2910 FPS	100 FPS
BALLISTIC DENSITY	NORMAL		100%	3.5%
BALLISTIC TEMPERATURE	NORMAL		59°F	25°F
BALLISTIC RANGE WIND	NORMAL		0 KTS	20 KTS
TIME-OF-FLIGHT (AA ONLY)	UNIFORM	0/38 SEC		
TARGET ALTITUDE	TRUNCATED NORMAL	-50/5000 FT	0 FT	2000 FT

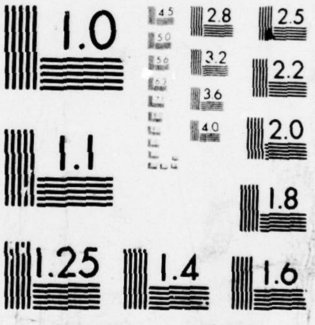
AD-A074 800 ARMY RESEARCH OFFICE RESEARCH TRIANGLE PARK NC F/G 5/9
PROCEEDINGS OF THE 1979 ARMY NUMERICAL ANALYSIS AND COMPUTERS C--ETC(U)
SEP 79

UNCLASSIFIED ARO-79-3 NL

2 OF 6

AD
A074 800





MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS-1963-A

OTT FUNCTIONS

- FK = FK(R2S, E2S, UM, D, T)
  25 TERM, 3rd DEGREE POLYNOMIAL
- N = 1 + .0001 R2S FK (2 + .0001 R2S FK)/3
- DELT = DELT(R2S, E2S, UM, D, T)
  13 TERM, 3rd DEGREE POLYNOMIAL
- DRS = DRS [T2(R2S, E2S), R2S, E2S]
  6 TERM
- RNG45 = RNG45(UM, D, T)
  7 TERM, 2nd DEGREE POLYNOMIAL

OTT FUNCTIONS

GUIDED PROJECTILE VERSION

- FK =FK(R2S, E2S, UM, D, T, MT)
  30 TERM, 3rd DEGREE POLYNOMIAL

- SAME

- DELT = DELT(R2S, E2S, UM, D, T, MT)
  18 TERM, 3rd DEGREE POLYNOMIAL

- NO DRIFT

- RNG45 = RNG45(UM, D, T, MT)
  8 TERM, 2nd DEGREE POLYNOMIAL

- RNG50 = RNG50(UM, D, T, MT)
  8 TERM, 2nd DEGREE POLYNOMIAL

- RNGMAX = RNGMAX(UM, D, T, MT)
  8 TERM, 2nd DEGREE POLYNOMIAL

CORE AND TIME COMPARISON

	LOCKHEED BALLISTICS MARK 41 AND MARK 48	OTT FUNCTIONS CONVENTIONAL ORDNANCE	OTT FUNCTIONS CONVENTIONAL AND GUIDED	OBRECHKOFF INTEGRATION CONVENTIONAL ORDNANCE
STORAGE	11433	10997	11393	8000*
EXECUTION TIME	124.9 MSEC	75.6 MSEC	83-85* MSEC	200* MSEC

*ESTIMATED

CHARGE	MK 48 ILLUM	MK 91 ILLUM	HF MK 393	HF MK 400	MK 41 VT,PD, AAC	MK 48 WP	SAL GUIDED
REDUCED	1	1	4	—	6	4	—
STANDARD	1	2	5	5	6	3	7
SUPER	—	—	6	6	—	—	—
TOTAL RANGE PARTITIONS	2	3	15	11	12	7	7
STORAGE LOCATIONS	88	132	681	498	542	322	336

TOTAL STORAGE = 2599

OTT FUNCTION ACCURACY

MARK 86 GUNFIRE CONTROL SYSTEM

PROJECTILE	CHARGE	RMS IN RANGE ERROR (MILS)
MARK 41	FULL	1.3
	REDUCED	2.4
MARK 48 ILLUM.	FULL	1.0
	REDUCED	.9
MARK 91 ILLUM.	FULL	1.7
MARK 48 WP	FULL	1.5
	REDUCED	1.8
HIFRAG 393	REDUCED	1.8
	FULL	.9
	SUPER	1.2
HIFRAG 400	FULL	1.4
	SUPER	1.3
GUIDED PROJECTILE		1.8
OVERALL		1.5

ADVANTAGES OF OTT FUNCTIONS

1. LESS COMPUTER STORAGE
2. FASTER EXECUTION TIME FOR BALLISTIC SOLUTION
3. NO INTERPOLATION IN AA MODE
4. NON-STANDARD VALUES OF IV, DENSITY, AIR TEMPERATURE, AND MOTOR TEMPERATURE ARE CONSIDERED DIRECTLY
5. INTERACTION EFFECTS OF IV, DENSITY, ETC. ARE INCLUDED IN SOLUTION
6. MANY FEWER PARTITIONS OF DATA IN BOTH AA AND SURFACE MODES
7. TARGET HEIGHT IS USED DIRECTLY IN SOLUTION INSTEAD OF ASSUMING RIGIDITY OF TRAJECTORY (SURFACE MODE)
8. FUNCTIONS CAN BE EXTENDED EASILY TO PROJECTILES OF DIFFERENT TYPES OR SIZES
9. COMMONALITY AMONG FIRE CONTROL SYSTEMS

REPRO-MODELING AND SIMULATION

Joseph T. Ryan/James L. Thompson
US Army Tank-Automotive Research & Development Command
Tank-Automotive Concepts Laboratory
Warren, Michigan 48090

ABSTRACT. As mathematical computer simulation models become large and complex, they become difficult to use; they are time consuming and expensive and are difficult to understand. This paper presents a simplified computational procedure of repro-modeling developed by Technology Service Corporation for the Tank-Automotive Research and Development Command. Repro-modeling is a technique for constructing an efficient input/output approximation to a complex model. A software package was developed to accomplish this. The procedure uses multi-variate linear regression to develop a continuous, multi-variate, piece-wise linear approximation to the relationship that exists between the input and output variables of the complex model. This paper discusses the software package and some of TARADCOM's experiences in applying it to analyses using existing computer models.

1. INTRODUCTION. As computer resources became more affordable and available over the last two decades, the size and complexity of computer-based simulations increased dramatically. For some types of applications, particularly those which involve the interfacing of several simulations, it is desirable or even essential to have simulations which are economical in their use of space and time within the computer. We encountered this problem at TARADCOM during the development of the Survivability Optimization Model. Attempts to simultaneously execute existing simulations of aspects of combat vehicle performance (such as mobility, detectability, etc.) revealed the need to be able to build a fast and compact "model of a model". We will describe here a technique, repro-modeling, for doing just that, and also describe TARADCOM's experience with COMPLIAR, a sofware package which uses this technique.

Repro-modeling is the creation of Dr. William Meisel of Technology Services Corporation, Santa Monica, California. The COMPLIAR software package was developed for TARADCOM by Technology Services Corporation. A more detailed description of the technique and its implementation in the software is contained in Reference 1.

2. Continuous Piecewise Linear Regression. The particular mathematical technique used in our applications of repro-modeling is multi-variate regression analysis using continuous piecewise linear functions. The basic scheme is this: Starting with the large and complex simulation, generate a data set of inputs and corresponding outputs. Fit to this data set an approximating function which is continuous everywhere, and linear in sub-regions of the input data space. This approximating function is the repro-model, and can be used to replace the original simulation. (Note that the scheme is still feasible if the data set is generated by the "real world", i.e., repro-modeling can be used to create a simulation model from field data.) It will become clear in what follows why continuous piecewise linear functions are an appropriate choice for the implementation of this scheme.

We must bear in mind that the objective in using repro-modeling is to produce an efficient input/output approximation to a complex model. We cannot expect, in any non-trivial case, that the approximation will retain all the precision of the original model. In addition, there is a practical limitation in that it is rarely practical to use the full set of input variables in the approximation; the current software will handle eight input variables at most; in practice, the cost of generating the data set and making the approximation becomes prohibitive as the number of input variables grows larger than this.

3. Continuous Piecewise Linear Functions. In general, a piecewise linear, real-valued function over a domain D in n-dimensional space can be represented as

$$F\ (\underline{x}) = F(x_1,\ldots x_n) = \sum_{j=1}^{n} b_{ij}\, x_j + b_i, \qquad \underline{x} \text{ in } X_i \qquad (1)$$

where X_i, i=1,...R are regions partitioning the domain D, and the b_{ij}, i=1,..,R, j=1,..., n , are constants.

At first sight, functions of this form have much to recommend them in this particular context. The form (1) involves fewer coefficients than other alternatives such as polynomials which do not make assumptions a priori on the shape of the approximation. As one extrapolates beyond the domain of the approximation, piecewise linear functions are not subject to the radical behavior characteristic of polynomials. However, there are many practical difficulties that must be overcome if an approximation of the form (1) is to be useful. The b_{ij} in (1) cannot be varied independently if the function is to remain continuous - a serious drawback in applying optimization schemes. Also, during the optimization process, the regions X_i themselves must be varied. A second type of problem involves the evaluation of the function itself. A prerequisite to the actual calculation of the value of F is the determination of the region X_i in which it lies. For a continuous piecewise linear function the X_i are bounded by n-1 dimensional planes. If n is greater than 1, the determination of the region X_i can be reduced to the evaluation of linear inequalities, but usually a large number of these will be involved. This process occupies far more time than the evaluation of F once the X_i is determined.

4. Efficient Representation of Piecewise Linear Functions. Here we set forth the scheme, due to Meisel, for representing continuous piecewise linear functions in a way which circumvents the difficulties alluded to above. The essential core of this scheme is the observation that convex continuous piecewise linear functions possess an especially simple representation.

As an example, consider the function F of one variable specified by

$$\begin{aligned} F(x) &= -1.5x + 9 \text{ for } x < 4 \\ &= 0.25x + 2 \text{ for } 4 \leq x < 10\ 2/3 \\ &= x - 6 \text{ for } x \geq 10\ 2/3 \end{aligned}$$

Graphing this function reveals that this definition is equivalent to

$$F(x) = \mathrm{Max}\ (-1.5x +9,\ 0.25x +2,\ x - 6),$$

a form which eliminates the determination of the regions as part of the process of evaluation of the function.

In the general case, we can represent any convex piecewise linear function in form

$$P\ (x_1,\ x_2, \ldots x_n) = \underset{i=1,2,\ldots,R}{\mathrm{Max}}\ \left(\sum_{j=1}^{n} a_{ij}\ x_j + a_i \right). \qquad (2)$$

The regions are now determined implicitly, and the a_{ij}'s can be varied arbitrarily without destroying the continuity of the function. Meisel coined the term "P-functions" for the representation of convex continuous piecewise linear functions. A general nonconvex continuous piecewise linear function can be represented by a weighted sum of P-functions,

$$F(x_1, \ldots, x_n) = \sum_{k=1}^{n} w_k P_k\ (x_1, \ldots, x_n), \qquad (3)$$

and it is this representation that is used in the COMPLIAR software package.

5. The COMPLIAR Software Package. COMPLIAR (Continuous Multi-variate Piecewise Linear Approximation and Regression) is a FORTRAN program which implements the idea of repromodeling using functions of the form (3). The program produces a continuous piecewise linear function which is the best approximation to the data set in the sense that it has the least mean square error. The optimization uses a gradient search algorithm.

In addition to carrying out the optimization process, COMPLIAR performs certain other functions essential to the efficient use of the technique. It accepts data from a file in practically any format and standardizes the data. It initialized the approximating function based on user input of the complexity desired, and carries out a three stage optimization progress. The parameters of the optimization (amount of time to be spent, various termination criteria, etc.) are under the control of the user. If the user desires, diagnostic information is output during the optimization progress. Upon termination of the optimization, the program provides statistical output which shows the success of the approximation in various subregions of the input data space. The capability to save the repro-model on a mass storage file, and to read this file in on a later run, is also included.

The optimization process is the standard gradient search least squares algorithm. The process proceeds through three stages, denoted as sequential, marginal, and joint optimization. The final stage is true least squares optimization of all the free parameters of the problem; the first two stages refine the initial choice of P-functions to insure that convergence in the last stage will be fairly rapid. A full description of the process is found in Reference 1.

6. Applications of Repro-modeling. The initial application of repro-modeling was to the AMC '71 mobility model. This is a large simulation which predicts the maximum cross-country speed possible across a given terrain in a given vehicle (wheeled or tracked). The full simulation uses over 80 parameters describing the vehicle and 22 parameters describing the terrain. Application of this model within the Survivability Optimization Model required a drastic reduction in the size of the code.

Using a combination of informed judgement and statistical analysis, we selected four vehicle and four terrain variables as input variables to the repro-model. The output variable was vehicle speed. With the remaining parameters set at average values for the situation under study, a data set of approximately 5200 points was generated using the AMC '71 model. We then used COMPLIAR to create a repro-model with 2 P-functions. In this case, each P-function consisted of two hyperplanes. The repro-model fit the data set extremely well (percent variation explained = 97%) and agreed with the original model over a broad range of the input variables, a range considerably larger than that used to generate the data set. The repro-model was 1/40 the size of the original model.

A second application of repro-modeling has been to create a continuous piecewise linear function representing the probability of hitting a combat vehicle as a function of the amount of the vehicle exposed, the range, and the velocity of the vehicle. These data were provided in tabular form by the Army Materiel Systems Analysis Activity. Here the purpose of the repro-model was to avoid a computationally costly table look-up and interpolation process. Here again, the software provided a satisfactory repro-model, although occasionally large errors have occurred at isolated points of the data set. In the cases where this has occurred, we have been able to achieve acceptable results by entering the offending data point into the data set repeatedly, effectively giving it greater weight.

7. Advantages and Limitations. In our experience, the techniques used in COMPLIAR constitute a powerful and reliable tool for data analyses. We have subjected the algorithm to a certain amount of testing, with good results. Given a null data set (output variable a random function of the inputs), the algorithm fails to converge. Typically the percent variation explained remains about 6% vs. 80-90% for a typical "good" fit. Given a data set where the output variable is a linear function of the inputs, the algorithm recovers the linear function.

The COMPLIAR package has, of course, its limitations. At present it is restricted to performing unweighted least squares optimization. In some cases, this is unsatisfactory in that large errors can occur on a few points while the mean square error remains small. We have, when necessary, entered critical data points into the data set a number of times, thus effectively weighting them. It would be comparatively easy to adapt the software to weighted least squares optimization.

It is difficult to make general statements about the effectiveness of any method of multi-variate regression, but from our experience the following picture has emerged. COMPLIAR does an exceptionally good job of fitting data sets which are essentially convex or concave. When the second gradient is indefinite, or is positive in some regions and negative in others, the method is less successful, probably because of the mean square error method used. Loosely speaking, it tends to clip the peaks and fill the valleys. To some extent this tendency can be overcome by increasing the complexity of the approximating function, but this increases the computation time required.

8. Availability of the Software. COMPLIAR is written in FORTRAN, and is currently implemented on the CDC 6500/6600 system at ARRADCOM, Dover, New Jersey. The source code is approximately 3000 lines long. The amount of core required to execute the program depends on the size of the data set, but moderate sized sets (about 1500 values) can be analyzed in a field length of 60000 (octal) words on the CDC. An IBM version of the source code is also available. The package is available without charge to interested Army and government users from either of the authors, at the following address: Commander, U. S. Army Tank-Automotive Research and Development Command, Attention: DRDTA-ZSS, Warren, Michigan 48090.

A NUMERICAL CHALLENGE - THE SOLUTION OF VARIATIONAL MODELS FOR MESOSCALE ATMOSPHERIC ANALYSIS

William D. Ohmstede
Battlefield Environment Division
US Army Atmospheric Sciences Laboratory
White Sands Missile Range, New Mexico

ABSTRACT. The kinematic and dynamic properties of atmospheric behavior are reviewed, especially in regard to the scales of disturbances and the effects of complex terrain. The utility of non-Euclidian geometries and the quasi-steady state assumption is described. The approach for the formulation of a quite general class of variational analysis models is developed. Both strong and weak constraints are considered. It is shown that problems of this type generally reduce to systems of elliptic-type partial differential equations. Besides the problems which are developed by use of the calculus of variations, a unique model involving direct variational adjustment is described.

The approaches used for numerical solution of the various models are discussed. There are no serious difficulties in solving several of the more elementary models since the direct-method algorithms of Sweet and/or Rosmond can be used. Over-relaxation is the most common method used but there are serious problems with this approach since convergence can be very slow or even nonexistent. Some tricks are described which can help stabilize and speed up the relaxation process. A challenge for the development of better methods is proffered. Several possibilities are discussed, including the iterated-direct method suggested by Rosmond and the method of multi-level adaptive solution advocated by Brandt. The conclusion is the question asking: What would be the most cost-effective approach for solving these problems?

1. INTRODUCTION. The atmosphere is in perpetual turmoil - for a variety of reasons. Most everyone is familiar with the highs and lows, cyclones and anticyclones, and frontal systems popularized by the newspaper weather maps and those of the TV weatherpersons. But these large scale disturbances are only a part of the picture since there is a broadband distribution of atmospheric perturbations as depicted in Figure 1, where the abscissa is the log length scale and the ordinate is the log velocity scale. The principal areas shown range from molecular unrest to the macroscale general circulation visualized by meteorological satellites. Also included are transient (M_T) and terrain-affected (M_S) mesoscale disturbances. The latter are of particular interest in this report.

Usually, the time scales of transient phenomena are roughly related to the length scale such that the dashed lines in Figure 1 provide a crude means of estimating the time scales. However, because the terrain is stationary and the associated adaptive processes have relatively small time scales, the mesoscale terrain effects can be assumed quasi-steady state, or at least to have time scales of the order of macroscale transient phenomena.

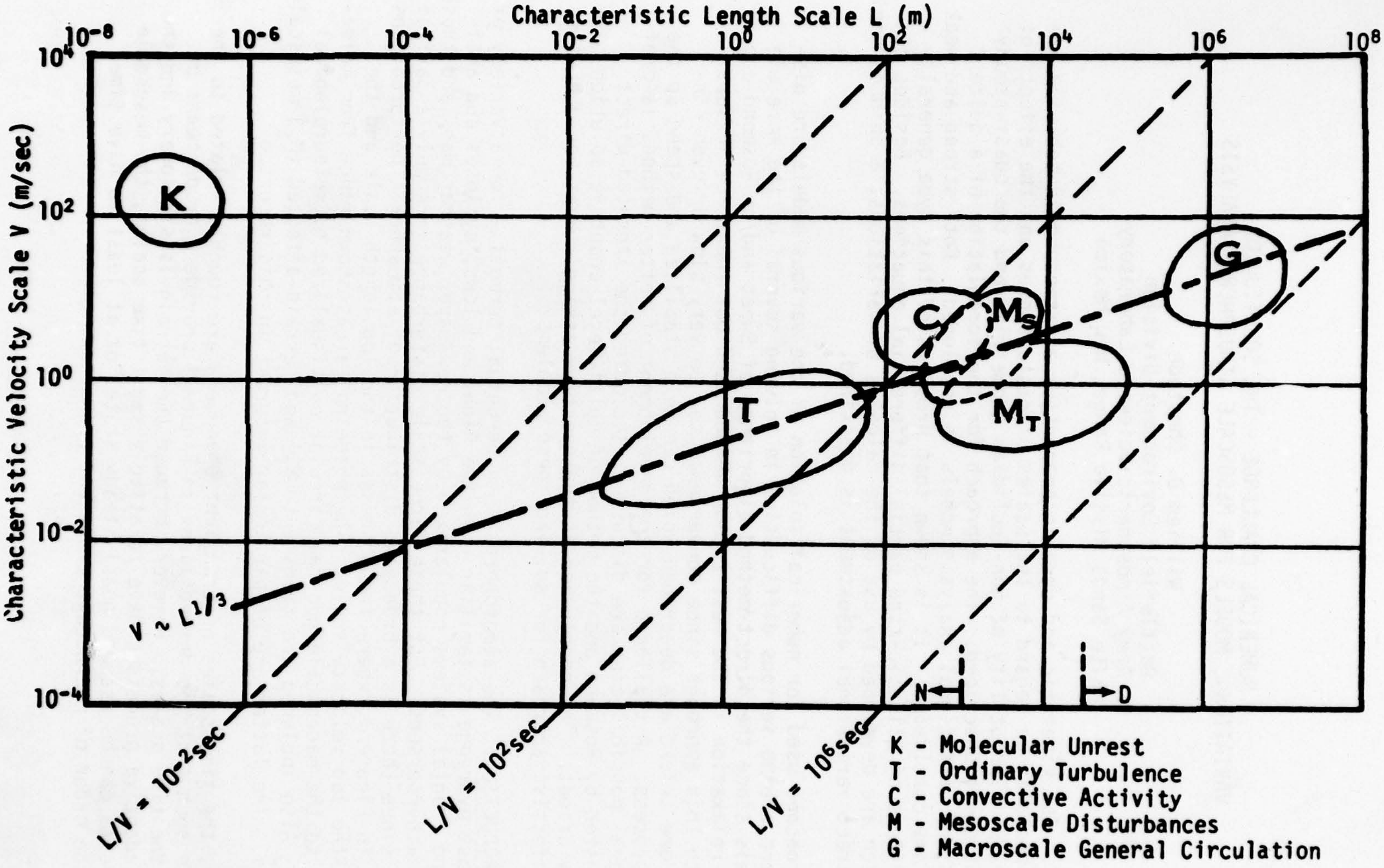


FIGURE 1. ATMOSPHERIC SCALE PROPERTIES (MODIFIED AFTER LETTAU)

The tick marks along the bottom of Figure 1 partition the disturbances into deterministic (D) and non-deterministic (N) phenomena, separated by a middle ground. Over the last decade, the US Army Atmospheric Sciences Laboratory has carried out research to develop numerical mesoscale atmospheric models which would reduce the length scale limit of determinism one or more orders of magnitude, at least in regard to terrain effects on the atmosphere.

What is meant by determinism is that we have the wherewithal to find deterministic solutions of problems which approximate atmospheric behavior at whatever scale is of interest. The approximations are meteorological as well as numerical. The Navier-Stokes equations, the first law of thermodynamics, and like equations describe atmospheric behavior, in principle, for all scales above molecular unrest. In practice, these equations must be messaged by averaging, filtering, or whatever to adapt them to the problem, and to effect closure.

2. BASIC PRECEPTS. The independent variables are time (t), and a Cartesian system of space coordinates ($\mathbf{x} \subset \mathbf{X}^3$) in a mesoscale domain from a few kilometers to several hundred. The kinematic variables are the deterministic atmospheric velocity components ($\mathbf{u} \subset \mathbf{U}^3$) commensurate with the space coordinates. The dependent variables ($\mathbf{y} \subset \mathbf{Y}^n$) are the unknown fields to be determined. Usually, the kinematic variables are a subset of the unknown dependent variables. The behavior of the dependent variables is generally governed by dynamic equations of the form

$$F^i \equiv \partial y^i/\partial t + \partial(y^i u^j)/\partial x^j + f^i(t,\mathbf{x},\mathbf{y}) = 0 \qquad (1)$$

In the case of the quasi-steady state approximation, the time derivatives are neglected or otherwise explicitly approximated. Boundary (and initial) conditions are required to complete the problem, and those at the air/earth interface are of paramount importance in the case of terrain effects.

There are two types of terrain effects on the atmosphere. The first is the case of mechanically created disturbances due to flow over complex terrain, and second are thermodynamically induced perturbations related to complex air/earth heat exchange patterns - a simple example being a sea breeze. In either case, we are dealing with boundary effects at the air/earth interface. Because of the significance of this irregular boundary, it is frequently useful to define a vertical coordinate conforming to the terrain, as illustrated in Figure 2. Here, h is the height of the ground, H is the elevation of an explicitly or implicitly determined upper boundary, and D is the layer thickness. The equation of Figure 2 is a sample transformed vertical coordinate of a system which generally would no longer be Euclidian. The corresponding transformation of the kinematic variables results in a terrain conforming velocity field. Furthermore, the transformed dynamic equations may pick up terrain effect forcing terms in the interior domain, not just at the boundary.

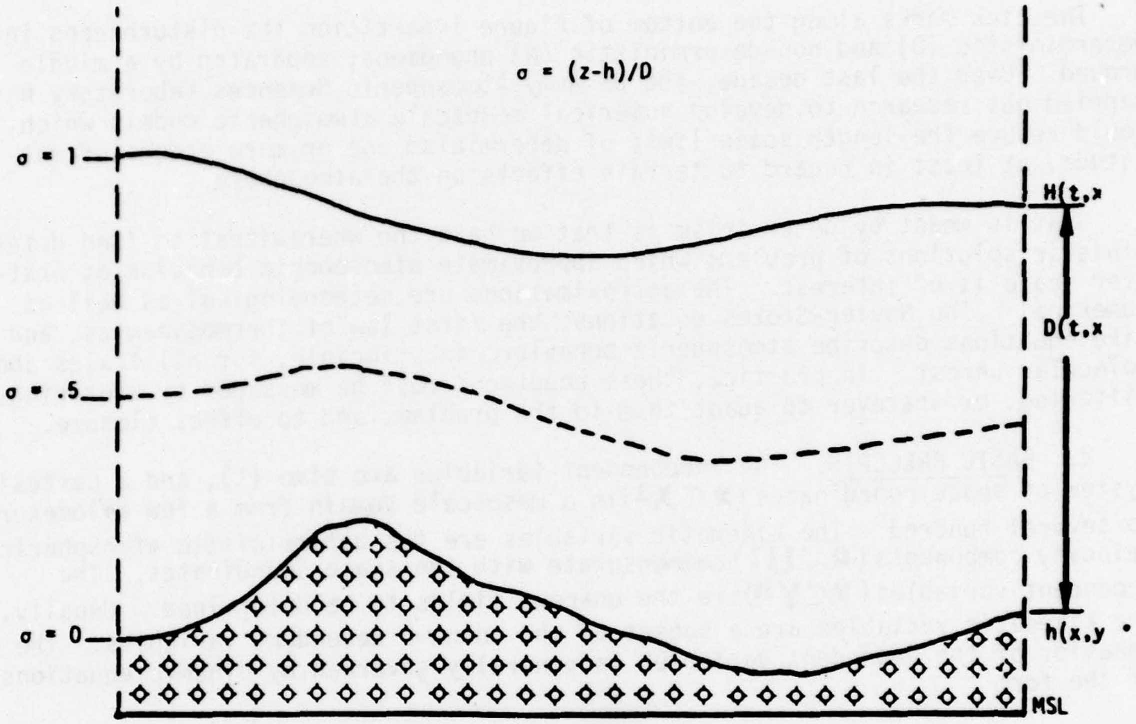


FIGURE 2. VERTICAL COORDINATE TRANSFORMATION

There are forecasts and then there are nowcasts. In the former case, a closed system of dynamic equations, along with appropriate boundary conditions, is solved as an initial value problem. Such is the twice daily activity of NOAA's National Meteorological Center, The USAF's Global Weather Central, and the Navy's Fleet Numerical Weather Central, when they independently prepare macroscale forecasts for the northern hemisphere. A nowcast, on the other hand, is a form of analysis which evaluates the current, rather than future, atmospheric state, especially on the mesoscale. In particular, limited atmospheric observations are combined with detailed terrain data as input to a mesoscale atmospheric terrain-effects model.

An ill-posed problem is likely if one combines an already closed system with additional constraints such as those imposed by observations. This and several other arguments have led to the development of atmospheric variational analysis techniques.

3. VARIATIONAL ANALYSIS. The calculus of variations has been around a long time. However, it was not until 1958 (ref. 1) that these concepts were proposed for use in meteorological analysis; and it has only been in the last five years that they have been applied to atmospheric terrain effects largely as a result of the ASL Mesometeorology research program.

Let us consider the concepts of the calculus of mesoscale variations. We begin by formulating a rather general integrand.

$$I \equiv \sum_i \kappa^i (y^i - \tilde{y}^i)^2 + \sum_j \lambda^j F^j + \sum_k \beta^k (F^k)^2 \qquad (2)$$

where the y^i are the variational solutions we seek relative to the initial guess fields $\tilde{y}^i$, and where the squared differences are essentially weighted by Gauss precision moduli (κ^i). We have thrown in mutually exclusive sets of strong constraints, ã la Lagrange, and weak constraints, respectively. Strong constraints (F^j) must be satisfied identically, whereas the weak constraints (F^k) are satisfied only in a least-squares sense. The κ^i and β^k are weights prescribed by the modeler but the Lagrange multipliers (λ^j) are unknown variables to be determined. The formulation of variational analysis problems is a rather loose process, but one thing for sure, the number of strong constraints must be less than the number of y^i dependent variables.

We integrate the integrand I over the mesoscale volume of interest.

$$\Phi \equiv \int_V I \cdot dV \qquad (3)$$

In accord with the calculus of variations, we want to minimize the integral Φ. Put in common sense terms, we want to find y fields which do not wander too far from the $\tilde{y}$ fields, in a least-squares sense, but which are consistent with the stated constraints. Obviously, $y = \tilde{y}$ if there are no constraints.

The reader should recall the Euler equation

$$\partial I/\partial y^i - \partial[\partial I/\partial y^i_\ell]/\partial x^\ell = 0 \qquad (4)$$

and boundary condition

$$n^\ell \cdot \partial I/\partial y^i_\ell |_S = 0 \qquad (5)$$

for each i, which result from the classical calculus of variations. Once these are established, one further differentiates and consolidates the Euler equations so as to drop out terms which combine to form the strong constraints. The results are what we call the model equations. Let us illustrate this with a toy example.

Consider a one-dimensional case with the continuity equation as a single strong constraint. This equation says that the local time rate of change of density (ρ) and the momentum ($m \equiv \rho u$) divergence sum to zero. The integrand for minimization is:

$$I = a(\rho_t - \tilde{\rho}_t)^2 + b(m - \tilde{m})^2 + \lambda(\rho_t + m_x).$$

Note that the time derivative of density is dealt with as just another variable. The Euler equations

$$2a(\rho_t - \tilde{\rho}_t) + \lambda = 0 \tag{6a}$$

$$2b(m - \tilde{m}) - \lambda_x = 0 \tag{6b}$$

and boundary condition,

$$\lambda = 0, \tag{7}$$

are quite straightforward. Finally, the model equation,

$$\lambda_{xx} - (b/a)\lambda = -2b[\tilde{\rho}_t + \tilde{m}_x], \tag{8}$$

is a linear Helmholz equation. In the limit as the ratio (b/a) goes to zero, we have a Poisson equation; which is the essence of the quasi-steady state assumption. The main purpose of this toy example is to illustrate that, by and large, the variational analysis problems are elliptic, particularly those with strong constraints.

4. VARIATIONAL SOLUTIONS. Most readers know of one or more ways to solve equation (8) so long as the inhomogeneous term is well behaved (which it is). Even an analytical solution may be possible. However, the problem becomes more difficult when one goes to two and three dimensions.

Generally, problems are discretized, even if they are linear. Furthermore, it is usual practice to resort to successive over-relaxation (SOR) to obtain solutions. The SOR approach is the cheap way out, from the programming standpoint, but it may not be for CPU time. Convergence may be quite slow or nonexistent. SOR's are especially bad with Neumann boundary conditions and complicated non-linear systems. Besides adjustments of the over-relaxation factor (sometimes under-relaxation is best for non-linear problems), there are steps one can take to improve SOR convergence. In one of the more complicated models, Stephens (ref. 2) introduced into the variational integrand an additional sum involving the weighted squared difference between the forthcoming and previous iterate of each dependent variable. Once the solution converges, these differences vanish so in theory they contribute nothing to the solution but aid the solution process. There are other tricks that can be used to help stabilize the SOR method but one must be careful that the problem is not altered so much that the solution attained is not actually the solution originally intended.

Now in fact, direct methods are available to solve elliptic problems which are separable. A general FORTRAN program, due to Sweet (ref. 3), can be obtained from the National Center for Atmospheric Research for solution of 2-D problems; and others, due to Rosmond (ref. 4), can be obtained from the US Navy Environmental Prediction Research Facility for 3-D problems. These programs use block cyclic matrix reduction which requires that the array dimensions be powers of two or other integers. These algorithms are tremendously powerful tools for solving separable elliptic problems.

Unfortunately, hardly any of the variational analysis models involve separable PDE's; indeed, most are non-linear. Even so, direct-solution advocates suggest that in this case one must iterate to obtain the solution. That is, terms which are non-separable are written as the sum of terms which are and terms which are not, the latter being estimated explicitly from prior iterates (or initial guesses) and combined with the inhomogeneous terms on the right-hand side of the equations. The resultant separable problem is solved repeatedly until residuals are annihilated.

There is yet another way to go. Brandt (ref. 5) has introduced the concept of multi-level adaptive solution of boundary-value problems. He states that SOR reduces high-frequency residual components but grinds interminably on low frequencies. His idea is to use SOR to knock out high frequency residuals, but when convergence becomes slow, switch to a coarser grid and again do SOR. Repeat the process. On the coarsest practical grid, SOR is continued until reasonable convergence is attained. Now, interpolate back to the next finest grid and try SOR again. Continue the process until convergence is attained on the finest grid. Claims are made that the multi-level method beats straight SOR all hollow.

The author has had years of adversity with SOR's but has not yet had experience with iterated-direct or multi-level methods. It is a case of a meteorologist sitting on a mathematical fence not knowing which way to fall. Actually, meteorologists seek solutions; if any mathematicians are in search of problems, please be our guest!

Finally, mention should be made of a unique, infra-microscale, terrain-effects model (ref. 6) we have based on Gauss's principle of least constraints in non-Euclidian coordinates. The streamlines shown in Figure 3 characterize the effect of rolling hills in central Germany on the near surface wind field as analyzed by this model. These are fascinating results, especially in regard to smoke munition deployment.

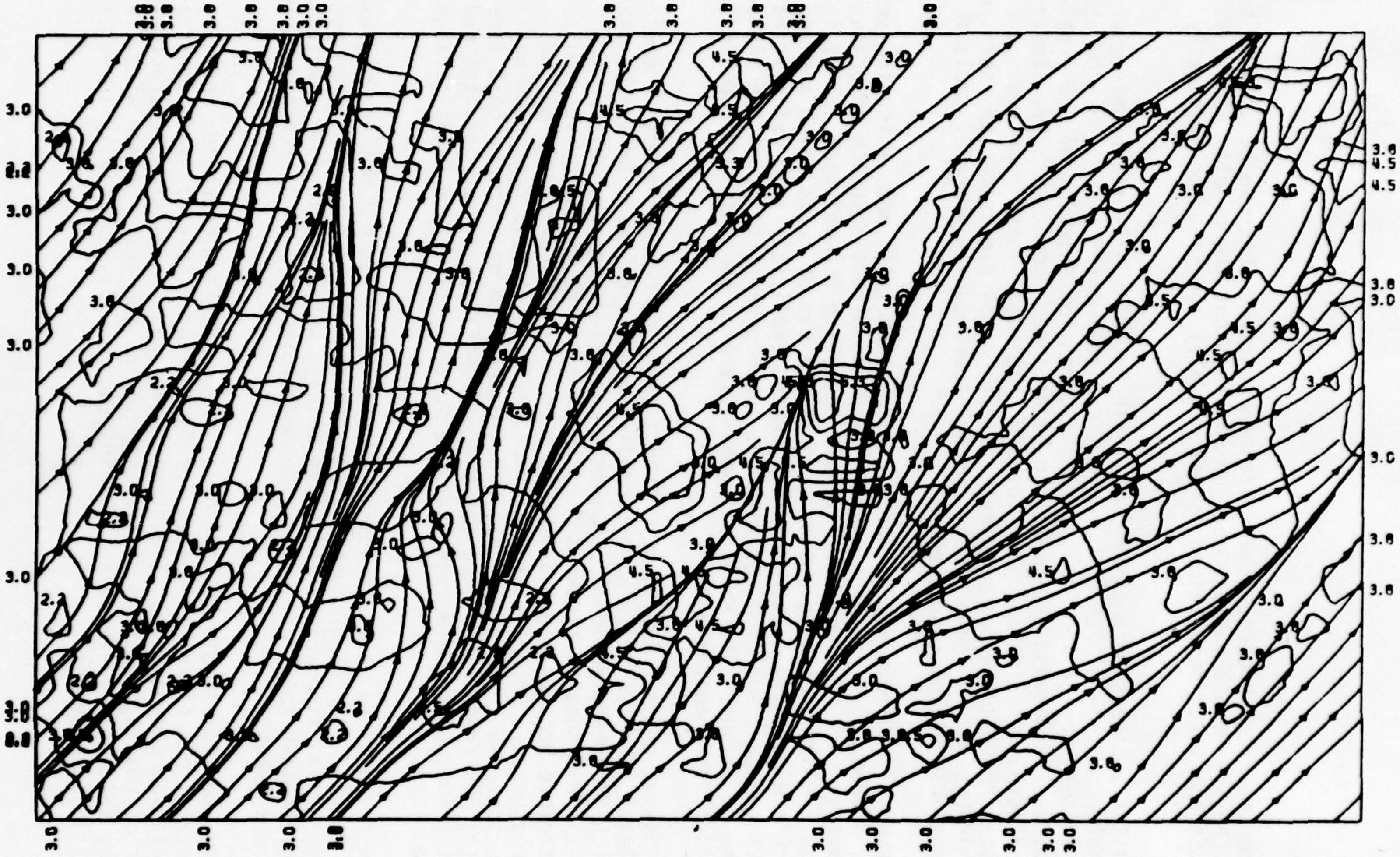


FIGURE 3. WIND FIELD - 3x5 KM COUNTRYSIDE PLAT EAST OF FULDA, GERMANY

This model is especially unique in that it foregoes the use of Euler equations. The integral is discretized directly and iteratively minimized. SOR works poorly on the problem. Indeed, one must severely under-relax just to keep from blowing up. Of the various variational analysis models, this is the one most desperately in need of a more satisfactory means of solution.

5. CONCLUSION. There is a quite general class of variational models for mesoscale atmospheric analysis which offer a major numerical challenge for the development of cost-effective methods of solution.

6. REFERENCES

1. Sasaki, Y., 1958: An Objective Analysis Based on the Variational Method. J. Meteor. Soc. Japan, 36, 77-88.

2. Stephens, J. J., and R. L. Inman, 1978: Mesoscale Diagnostic Numerical Variational Analysis Models, Final Report, Contract DAAD07-76-C-0037, Chap 11.

3. Sweet, R. A., 1973: A Generalized Cyclic Reduction Algorithm. SIAM J. Num. Anal., 10, 506-520.

4. Rosmond, T. E., 1975: Subroutines for Solving Three-Dimensional Elliptic Equations. EPRF Computer Programming Note No. 22. Naval Environmental Prediction Research Facility, Monterey, California, 39 pp.

5. Brandt, A., 1977: Multi-Level Adaptive Solutions of Boundary-Value Problems. Math. Comp., 31, 333-390.

6. Ball, J. A., and S. A. Johnson, 1978: Physically-Based High Resolution Surface Wind and Temperature Analysis for EPAMS, Final Report, Contract DAEA18-77-C-0043.

A FORTRAN ROUTINE FOR ESTIMATING NORMAL DISTRIBUTION PARAMETERS

Bernard N. Goulet
U.S. Army Materiel Systems Analysis Activity
Aberdeen Proving Ground, Maryland 21005

ABSTRACT. A Monte Carlo computer program simulating engagement of a single target by a tank main armament system has been developed by the Joint Munitions Effectiveness Manual Methodology and Evaluations Working Group. Exercise of this program yields large amounts of data concerning the location of round impacts in the target plane. This paper describes a computer routine for processing the horizontal or vertical coordinates of hitting or missing rounds to obtain an estimate of the mean and three estimates of the standard deviation for a normal distribution tentatively assumed to fit the data. After the computer run, an analyst can judge by comparing the three sets of parameters the extent to which a normal distribution applies and make a best estimate for the parameters of interest.

1. INTRODUCTION.

A Monte Carlo computer program simulating the engagement of a single target by a tank main armament system has been developed by the Joint Munitions Effectiveness Manual (JMEM) Methodology and Evaluations Working Group, a tri-service group responsible for establishing certain standardized estimates of weapon effectiveness. The program is often referred to as the JMEM Direct Fire program. Exercise of this program yields large amounts of data concerning the location of round impacts in the target plane. First rounds fired against a particular target are of chief concern, and these can be subdivided into rounds that hit and rounds that miss the target. Chart 1 illustrates a possible target and a few conceivable first round impact points.

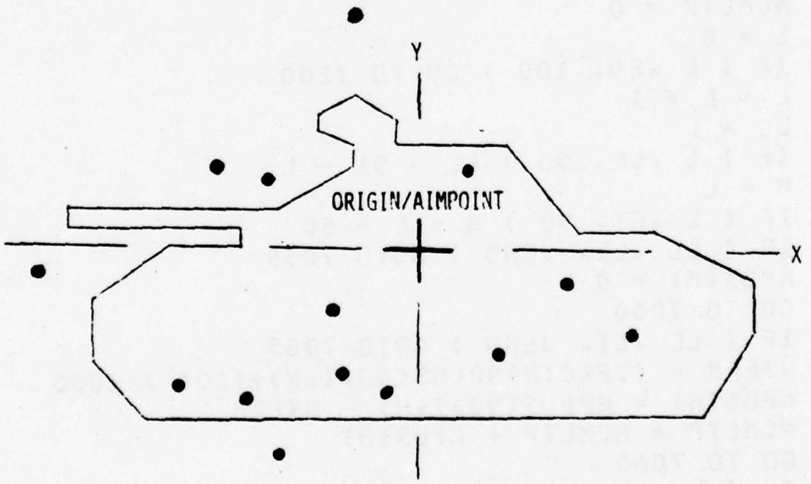


CHART 1 IMPACT POINTS FOR HITTING AND MISSING FIRST ROUNDS

CHART 2 PROGRAM STATEMENTS

```
     DO 7010 I = 1,2
     AVRG = 0.0
     TMINUS = 0.0
     TPLUS = 0.0
     KTIMES = 0
     DO 7020 J = 1,50
     TERM1 = NMINUS(J,I,N)
     TERM2 = NPLUS(J,I,N)
     TMINUS = TMINUS + TERM1
7020 TPLUS = TPLUS + TERM2
     TOTAL(I) = TMINUS + TPLUS
     NTOTAL = TOTAL(I)
     T5000 = TOTAL(I) * 0.5
     N5000 = T5000
     IF ( T5000 .LT. TMINUS ) GO TO 7025
     DIFF = T5000 - TMINUS
     DO 7030 J = 1,50
     PLUS = NPLUS(J,I,N)
     IF ( PLUS .LT. DIFF ) GO TO 7035
     IFRCTN = 1000.0 * DIFF / PLUS
     FRCTN = FLOAT(IFRCTN) / 1000.0
     AVRG = AVRG + FRCTN*20.0
     JEND = 100 - KTIMES
     LTIMES = 50 - KTIMES
     LEND = 1 + KTIMES
     DO 7040 K = 1,50
     NNEG(K) = 0
7040 NPOS(K) = 0
     NCMLTN = 0
     NCMLTP = 0
     L = 0
7050 IF ( L .EQ. 100 ) GO TO 7200
     L = L + 1
     LL = L
     IF ( L .LE. 50 ) LL = 51 - L
     M = L
     IF ( L .GT. 50 ) M = L - 50
     IF ( LL .LE. JEND ) GOTO 7055
     NPOS(M) = 0
     GO TO 7060
7055 IF ( LL .LT. JEND ) GOTO 7065
     NTERM = (IFRCTN*NPLUS(50,I,N)+500) / 1000
     NPOS(M) = NPLUS(50,I,N) - NTERM
     NCMLTP = NCMLTP + NPOS(M)
     GO TO 7060
7065 IF ( LL .LE. LTIMES ) GOTO 7075
     IF ( LL .LE. 50 ) GOTO 7070
     NTERM1 = (IFRCTN*NPLUS(L-LTIMES,I,N)+500) / 1000
     NTERM2 = (IFRCTN*NPLUS(L-LTIMES+1,I,N)+500) / 1000
     NPOS(M) = NPLUS(L-LTIMES,I,N) - NTERM1 + NTERM2
     NCMLTP = NCMLTP + NPOS(M)
     GO TO 7060
```

```
7070 NTERM1 = (IFRCTN*NPLUS(LEND-M,I,N)+500) / 1000
     NTERM2 = (IFRCTN*NPLUS(LEND-M+1,I,N)+500) / 1000
     NNEG(M) = NPLUS(LEND-M,I,N) - NTERM1 + NTERM2
     NCMLTN = NCMLTN + NNEG(M)
     GO TO 7060
7075 IF ( LL .LT. LTIMES ) GOTO 7085
     NTERM1 = (IFRCTN*NMINUS(1,I,N)+500) / 1000
     NTERM2 = (IFRCTN*NPLUS(1,I,N)+500) / 1000
     NNEG(M) = NMINUS(1,I,N) - NTERM1 + NTERM2
     NCMLTN = NCMLTN + NNEG(M)
     GO TO 7060
7085 IF ( LL .EQ. 1 ) GO TO 7095
     NTERM1 = (IFRCTN*NMINUS(LTIMES-LL+1,I,N)+500) / 1000
     NTERM2 = (IFRCTN*NMINUS(LTIMES-LL,I,N)+500) / 1000
     NNEG(L-KTIMES+1) = NMINUS(LTIMES-LL+1,I,N) - NTERM1 + NTERM2
     NCMLTN = NCMLTN + NNEG(L-KTIMES+1)
     GO TO 7060
7095 NNEG(50) = (IFRCTN*NMINUS(49-KTIMES,I,N)+500) / 1000
     NXTRA = KTIMES + 1
     DO 7100 JXTRA = 1,NXTRA
7100 NNEG(50) = NNEG(50) + NMINUS(49-KTIMES+JXTRA,I,N)
     NTERM = (IFRCTN*NMINUS(50,I,N)+500) / 1000
     NNEG(50) = NNEG(50) + NTERM
     NCMLTN = NCMLTN + NNEG(50)
7060 IF ( NCMLTP .EQ. NTOTAL ) L = 100
     IF ( L .GT. 50 ) GO TO 7050
     IF ( NCMLTN .LT. N5000 ) GO TO 7050
     NCMLTP = NCMLTN
     L = 50
     GO TO 7050
7035 DIFF = DIFF - PLUS
     KTIMES = KTIMES + 1
     AVRG = AVRG + 20.0
7030 CONTINUE
7025 DIFF = TMINUS - T5000
     DO 7110 J = 1,50
     AMINUS = NMINUS(J,I,N)
     IF ( AMINUS .LT. DIFF ) GO TO 7115
     IFRCTN = 1000.0 * DIFF / AMINUS
     FRCTN = FLOAT(IFRCTN) / 1000.0
     AVRG = AVRG - FRCTN*20.0
     JEND = 100 - KTIMES
     LTIMES = 50 - KTIMES
     LEND = 1 + KTIMES
     DO 7120 K = 1,50
     NPOS(K) = 0
7120 NNEG(K) = 0
     NCMLTP = 0
     NCMLTN = 0
     L = 0
7130 IF ( L .EQ. 100 ) GO TO 7200
     L = L + 1
     LL = L
```

CHART 2 (CONTINUED)

```
      IF ( L .LE. 50 ) LL = 51 - L
      M = L
      IF ( L .GT. 50 ) M = L - 50
      IF ( LL .LE. JEND ) GOTO 7135
      NNEG(M) = 0
      GO TO 7140
7135  IF ( LL .LT. JEND ) GOTO 7145
      NTERM = (IFRCTN*NMINUS(50,I,N)+500) / 1000
      NNEG(M) = NMINUS(50,I,N) - NTERM
      NCMLTN = NCMLTN + NNEG(M)
      GO TO 7140
7145  IF ( LL .LE. LTIMES ) GOTO 7155
      IF ( LL .LE. 50 ) GOTO 7150
      NTERM1 = (IFRCTN*NMINUS(L-LTIMES,I,N)+500) / 1000
      NTERM2 = (IFRCTN*NMINUS(L-LTIMES+1,I,N)+500) / 1000
      NNEG(M) = NMINUS(L-LTIMES,I,N) - NTERM1 + NTERM2
      NCMLTN = NCMLTN + NNEG(M)
      GO TO 7140
7150  NTERM1 = (IFRCTN*NMINUS(LEND-M,I,N)+500) / 1000
      NTERM2 = (IFRCTN*NMINUS(LEND-M+1,I,N)+500) / 1000
      NPOS(M) = NMINUS(LEND-M,I,N) - NTERM1 + NTERM2
      NCMLTP = NCMLTP + NPOS(M)
      GO TO 7140
7155  IF ( LL .LT. LTIMES ) GOTO 7165
      NTERM1 = (IFRCTN*NPLUS(1,I,N)+500) / 1000
      NTERM2 = (IFRCTN*NMINUS(1,I,N)+500) / 1000
      NPOS(M) = NPLUS(1,I,N) - NTERM1 + NTERM2
      NCMLTP = NCMLTP + NPOS(M)
      GO TO 7140
7165  IF ( LL .EQ. 1 ) GO TO 7175
      NTERM1 = (IFRCTN*NPLUS(LTIMES-LL+1,I,N)+500) / 1000
      NTERM2 = (IFRCTN*NPLUS(LTIMES-LL,I,N)+500) / 1000
      NPOS(L-KTIMES+1) = NPLUS(LTIMES-LL+1,I,N) - NTERM1 + NTERM2
      NCMLTP = NCMLTP + NPOS(L-KTIMES+1)
      GO TO 7140
7175  NPOS(50) = (IFRCTN*NPLUS(49-KTIMES,I,N)+500) / 1000
      NXTRA = KTIMES + 1
      DO 7180 JXTRA = 1,NXTRA
7180  NPOS(50) = NPOS(50) + NPLUS(49-KTIMES+JXTRA,I,N)
      NTERM = (IFRCTN*NPLUS(50,I,N)+500) / 1000
      NPOS(50) = NPOS(50) + NTERM
      NCMLTP = NCMLTP + NPOS(50)
7140  IF ( NCMLTN .EQ. NTOTAL ) L = 100
      IF ( L .GT. 50 ) GO TO 7130
      IF ( NCMLTP .LT. N5000 ) GO TO 7130
      NCMLTN = NCMLTP
      L = 50
      GO TO 7130
7115  DIFF = DIFF - AMINUS
      KTIMES = KTIMES + 1
      AVRG = AVRG - 20.0
7110  CONTINUE
```

CHART 2 (CONTINUED)

```
7200 CONTINUE
     DO 7310 K = 1,50
     IF ( K .GT. 1 ) GO TO 7315
     NSUM1 = NPOS(K)
     NSUM2 = NNEG(K)
     GO TO 7320
7315 NSUM1 = NSUM1 + NPOS(K)
     NSUM2 = NSUM2 + NNEG(K)
7320 NCMPOS(K) = NSUM1
     NCMNEG(K) = NSUM2
     NRFLCT(K) = NSUM1 + NSUM2
7310 CONTINUE
     DO 7330 J = 1,50
     SIGMAX = J * 20
     IF ( J .GT. 1 ) GO TO 7335
     SMFRQ1 = 0.0
     SMFRQ2 = NRFLCT(1) * 10000 / NTOTAL
     K5 = 5
     K10 = 10
     GO TO 7350
7335 SMFRQ1 = SMFRQ2
     IF ( J .EQ. 50 ) GO TO 7345
     SMFRQ2 = NRFLCT(J) * 10000 / NTOTAL
     GO TO 7350
7345 SMFRQ2 = 10000.0
7350 FRQNCY = SMFRQ2 - SMFRQ1
     IF ( K5 .NE. 5 ) GO TO 7355
     IF ( SMFRQ2 .LT. 3829.2 ) GOTO 7330
     D05 = (SMFRQ2-3829.2) / FRQNCY
     SIG05 = (SIGMAX-D05*20.0) / 0.5
     K5 = 0
7355 IF ( K10 .NE. 10 ) GO TO 7365
     IF ( SMFRQ2 .LT. 6826.8 ) GOTO 7330
     D10 = (SMFRQ2-6826.8) / FRQNCY
     SIG10 = SIGMAX - D10*20.0
     K10 = 0
7365 IF ( SMFRQ2 .LT. 8663.8 ) GOTO 7330
     D15 = (SMFRQ2-8663.8) / FRQNCY
     SIG15 = (SIGMAX-D15*20.0) / 1.5
     GO TO 7370
7330 CONTINUE
7010 CONTINUE
```

(END OF CHART 2)

This paper describes a computer routine for processing the horizontal coordinates or the vertical coordinates of hitting rounds or of missing rounds. The routine provides, for each set of data considered, an estimate of the mean and three estimates of the standard deviation for a normal distribution tentatively assumed to fit the data. After the computer run, an analyst can judge by comparing the three standard deviation values whether the tentative assumption of normality is sufficiently substantiated and, if so, make a best estimate for the parameters of interest. The word "routine" is used for the logical processing documented in this paper. However, the associated program statements have not actually been structured as a separate routine, but are a portion of the complete engagement simulation program previously mentioned. A separate routine could readily be developed for other applications.

2. DESCRIPTION OF ROUTINE. The program instructions of interest are listed in Chart 2. Note that the entire chart consists of the loop DO 7010 I = 1,2. The index I equals 1 for horizontal coordinates and 2 for vertical coordinates. For any other application, one could allow for more values of I or establish a single dummy setting for this index.

J				N = 1	N = 1	N = 3	N = 3
				I = 1	I = 2	I = 1	I = 2
				(205)	(331)	(193)	(102)
8			NMINUS (J, I, N)	0	0	0	0
7				0	0	0	0
6				0	0	0	3
5				2	1	0	14
4				11	17	12	29
3				34	50	21	19
2				65	118	59	17
1	INTERVAL (INCHES)			93	145	101	20
1		0-20	NPLUS (J, I, N)	140	109	96	71
2		20-40		100	71	81	104
3		ETC		55	17	54	104
4				21	1	35	58
5				8	0	8	19
6				0	0	2	11
7				0	0	2	2
8				0	0	0	0
				(324)	(198)	(278)	(369)
				(529)	(529)	(471)	(471)

CHART 3 SAMPLE INPUT DATA

The index N is set before the DO 7010 loop and presently has possible values of 1 through 5. N equals 1 when hitting first rounds are of concern and 3 for missing first rounds. Meanings of the values 2, 4, and 5 need not be explained here. The range of values of N can, for other applications, be increased or decreased; in particular, a single dummy setting can be used.

The horizontal and vertical coordinate axes are each subdivided into 100 intervals of 20 inches each; positive and negative coordinates are each covered by 50 intervals. Intervals are related to the index J in a way that should soon be clear. Assume that one has run many simulated engagements and determined, for hitting rounds and missing rounds separately, how many times a particular interval contained the horizontal coordinate of the first round and, again separately, the vertical coordinate. Input data of this sort are shown in Chart 3. Numbers in the array NPLUS(J,I,N) indicate how many coordinates are within the interval 0 to 20 inches for J = 1, within the interval 20 to 40 inches for J = 2, and so forth. Similarly, the NMINUS(J,I,N) array contains the number of coordinates within the interval -20 to 0 inches for J = 1, and so forth.

The arrays involved in Chart 2 and their dimensions in the JMEM Direct Fire program are as follows:

```
NMINUS(50,2,5)
NPLUS(50,2,5)
TOTAL(2)
NNEG(50)
NPOS(50)
NCMPOS(50)
NCMNEG(50)
NRFLCT(50)
```

The NMINUS and NPLUS arrays contain the data to be processed.

The complete calculations for particular values of N and I yield an estimate of the mean, denoted by AVRG, and three estimates of the standard deviation, represented by SIG05, SIG10, and SIG15.

A total of 1000 engagements were simulated to obtain the data in Chart 3. A first round hit was obtained in 529 instances, and the first round missed the target in the other 471 instances. The impact points for the 529 hits were (for an observer at the firing weapon) to the right of the Y-axis illustrated in Chart 1 on 324 occasions, and to the left on the other 205 occasions. These same impact points were above the X-axis 198 times and below 331 times. Impact points of the 471 misses had positive horizontal coordinates in 278 engagements, and positive vertical coordinates in 369 engagements. To illustrate the remainder of these explanations, an arbitrary choice has been made of the values N = 3 (missing first rounds) and I = 2 (vertical coordinates).

It is useful to digress and consider how the mean and standard deviation of a normal distribution corresponding to the input data selected can be estimated graphically. Chart 4 shows the associated quantitative basis and Chart 5 the plotted points to which one would attempt, using judgement rather than calculation, to fit a straight line. The computer routine being described performs computations that basically parallel the graphical approach.

N=3, I=2

J			CUMULATIVE SUM	CUMULATIVE FRACTION (SUM/471)
6	NMINUS (J, I, N)	3	3	.006
5		14	17	.04
4		29	46	.10
3		19	65	.14
2		17	82	.17
1		20	102	.22
1	NPLUS (J, I, N)	71	173	.37
2		104	277	.59
3		104	381	.81
4		58	439	.93
5		19	458	.97
6		11	469	.996
7		2	471	1.000

CHART 4 BASIS FOR CHART 5

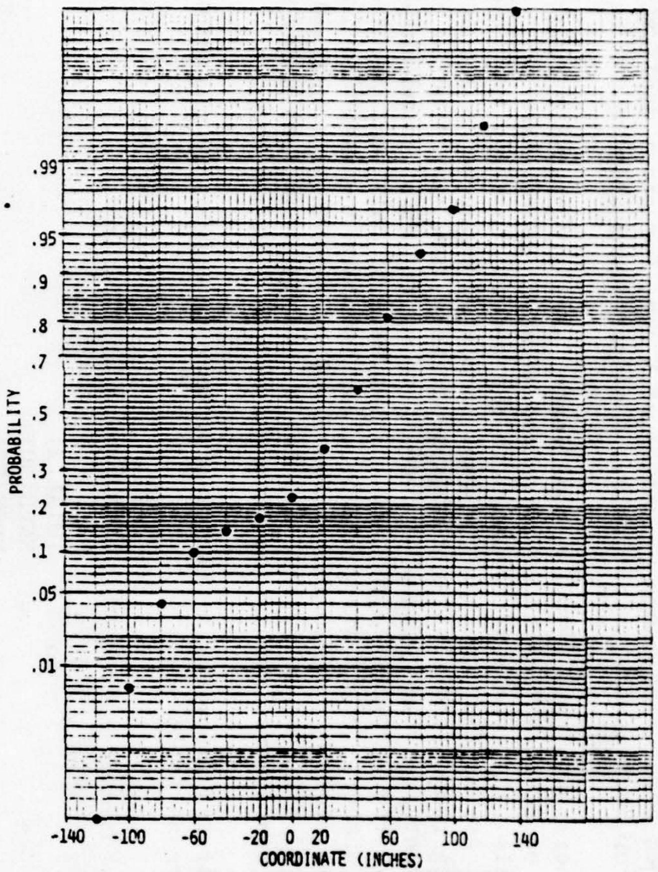
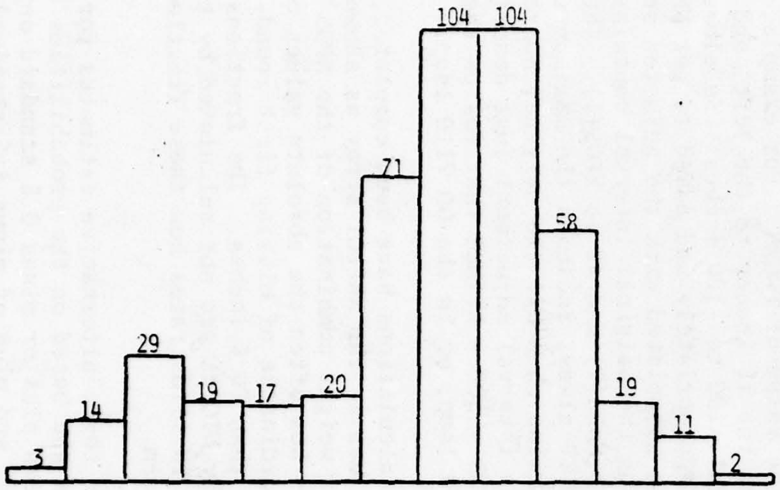


CHART 5 DISTRIBUTION OF COORDINATE DATA



```
TMINUS = 102.                           TPLUS = 369.

              NTOTAL = 471
              N5000  = 235
              DIFF   = 133.

      71. < 133.                        KTIMES = 1

              DIFF   = 62.

      104. > 62.                        FRCTN = .6

              AVRG   = 32.0
```

CHART 6 CALCULATION OF MEAN

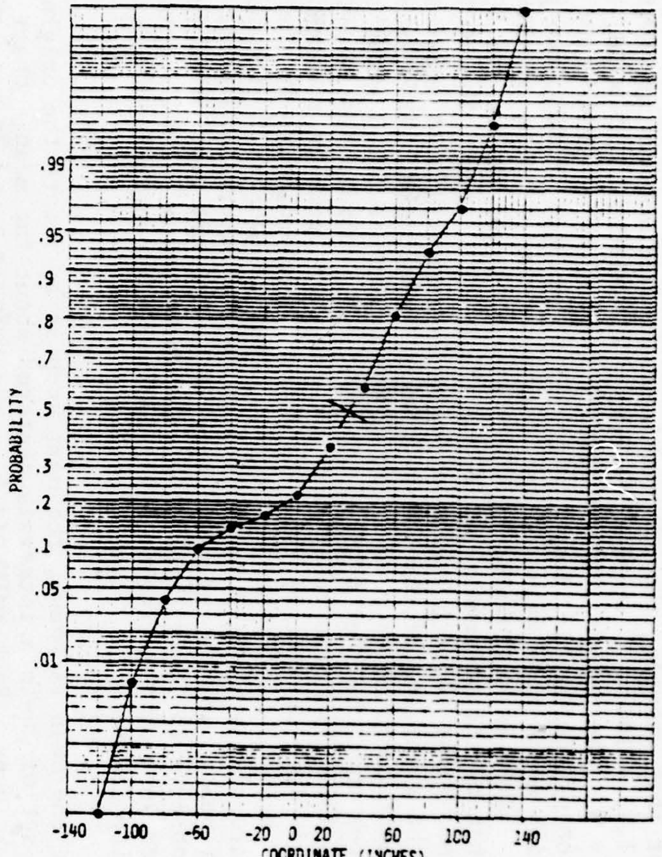


CHART 7 ESTIMATED MEAN (AVRG)

The steps involved in estimating the mean are illustrated in Chart 6. Note that, for this example, the nonzero input data of concern are NPLUS(1,I,N) = 71, NPLUS(2,I,N) = 104, . . ., NPLUS(7,I,N) = 2, NMINUS(1,I,N) = 20, . . ., NMINUS (6,I,N) = 3. The totals 102 and 369 are the values of TMINUS and TPLUS after completion of the DO 7020 loop. Since TPLUS exceeds TMINUS, the program attempts to identify a positive value on the Y-axis as the estimate of the mean. Consequently, DIFF is calculated by the statement right before the DO 7030 loop, rather than by the statement preceding the DO 7110 loop. The DO 7030 loop establishes first that the point of central tendency is at least as great as 20 inches, the right end of the first interval considered. This is so because 133 impact point coordinates need to be dropped from the NPLUS group before the coordinates equal half of NTOTAL. Since the 71 coordinates in NPLUS(1,I,N) are less than 133, KTIMES is reset to 1 and the difference is reduced to 62. Next, the DO 7030 loop determines that .6 of the 104 coordinates in the interval 20 to 40 inches need to be dropped. Since .6 times 20 equals 12, the points to be dropped are simply assumed to be located in the interval 20 to 32 inches, while all other points in the interval 20 to 40 inches are considered to exceed 32. One can observe in Chart 7 that the calculated value of AVRG corresponds to the point where one of the line segments joining adjacent points crosses the .5 probability level.

Once the mean is known, the next step is to relate the original input data to a new set of 20-inch intervals centered about the mean. Chart 8 illustrates how this is done. Each NMINUS(J,I,N) and NPLUS(J,I,N) value is first subdivided into two subelements according to the value of FRCTN. For example, NMINUS(6,I,N) is broken up into 2, associated with the 12 inches to the left, and 1, considered in the right 8 inches of the interval -100 to -120 inches. Subelements from adjacent intervals are then paired appropriately and added to get NNEG(M) and NPOS(M) values, where the index M is associated with the adjusted set of intervals. Note how NPLUS(2,I,N), which involves the original interval containing the estimated mean 32, contributes 62 to NNEG(1) and 42 to NPOS(1). The cumulative totals MCMLTN and NCMLTP, where NCMLTP always includes the maximum value of NCMLTN, enable the computer to determine when all the original nonzero input information has been processed. The interval adjustment just described is done by the statements beginning with JEND = 100 - KTIMES that follow the final determination of AVRG in the DO 7030 loop, or in the DO 7110 loop.

After the interval adjustment calculations have been completed, the DO 7310 loop of the program computes the values in the NRFLCT array as shown in Chart 9. The NRFLCT array represents an equal weight combination of the NPOS and NNEG data. Any NRFLCT(K) value indicates how often the absolute values of the differences between the vertical coordinates of missing first rounds and the estimated mean are equal to or less than 20 K inches. The fractions obtained when one divides the NRFLCT values by NTOTAL are not calculated by the program, but are included in Chart 9. Chart 10 illustrates how these fractions are related to the distribution of concern.

The DO 7330 loop determines the three alternative estimates for the standard deviation. These estimates are based on the probabilities associated in a normal distribution with the mean plus or minus 0.5 standard deviation, plus or minus 1.0 standard deviation, and plus or minus 1.5 standard deviation. Linear interpolation is applied, as necessary, to the distribution implied by the NRFLCT values to infer estimates of 0.5, 1.0, and 1.5 times the standard

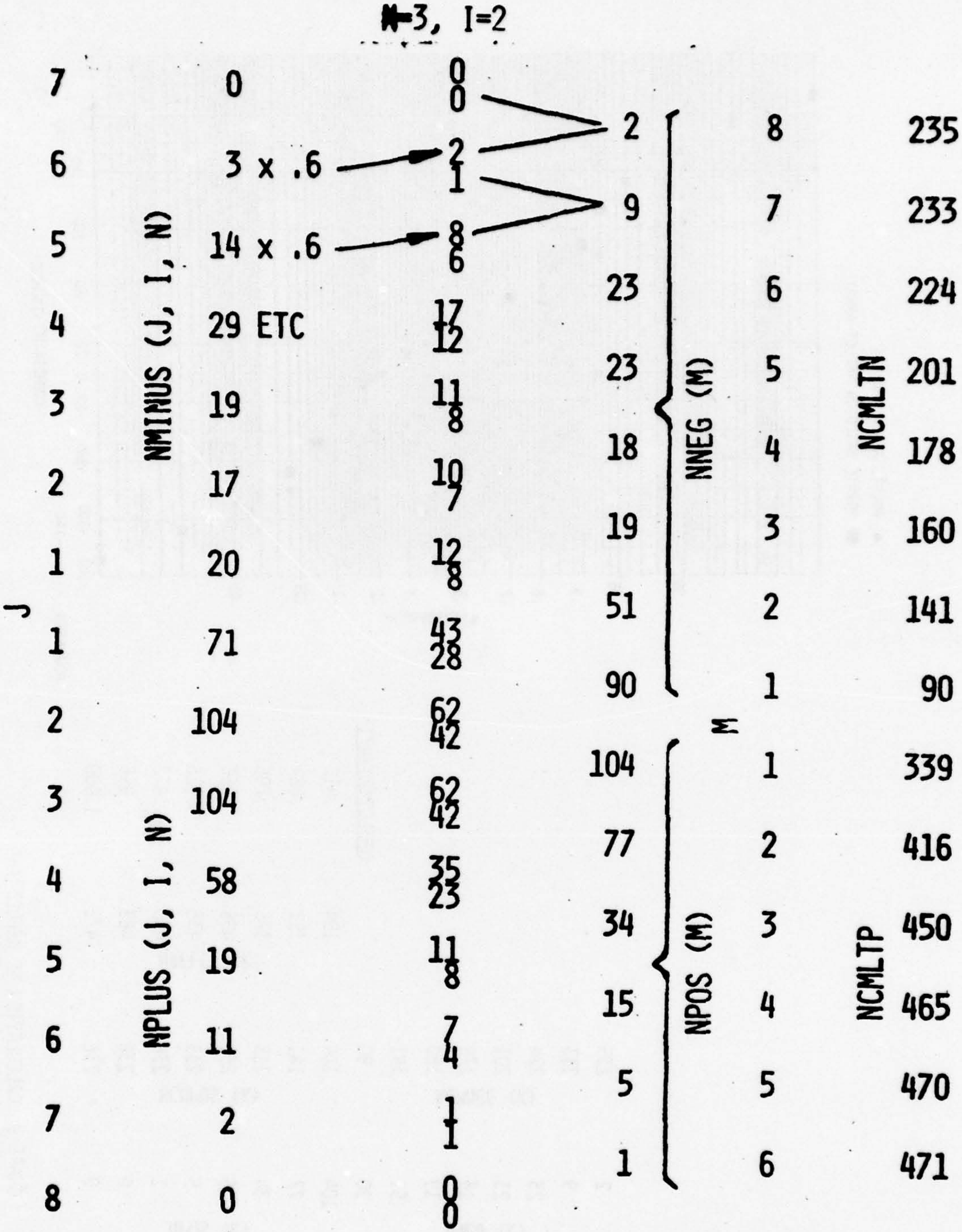


CHART 8 ADJUSTMENT OF INTERVALS

K	NNEG (K)	NCMNEG (K)
8	2	235
7	9	233
6	23	224
5	23	201
4	18	178
3	19	160
2	51	141
1	90	90

K	NPOS (K)	NCMPOS (K)	NRFLCT (K)	NRFLCT(K)/471
1	104	104	194	.412
2	77	181	322	.684
3	34	215	375	.796
4	15	230	408	.866
5	5	235	436	.926
6	1	236	460	.977
7	0	236	469	.996
8	0	236	471	1.000

CHART 9 CALCULATION OF NRFLCT AND CORRESPONDING FRACTIONS

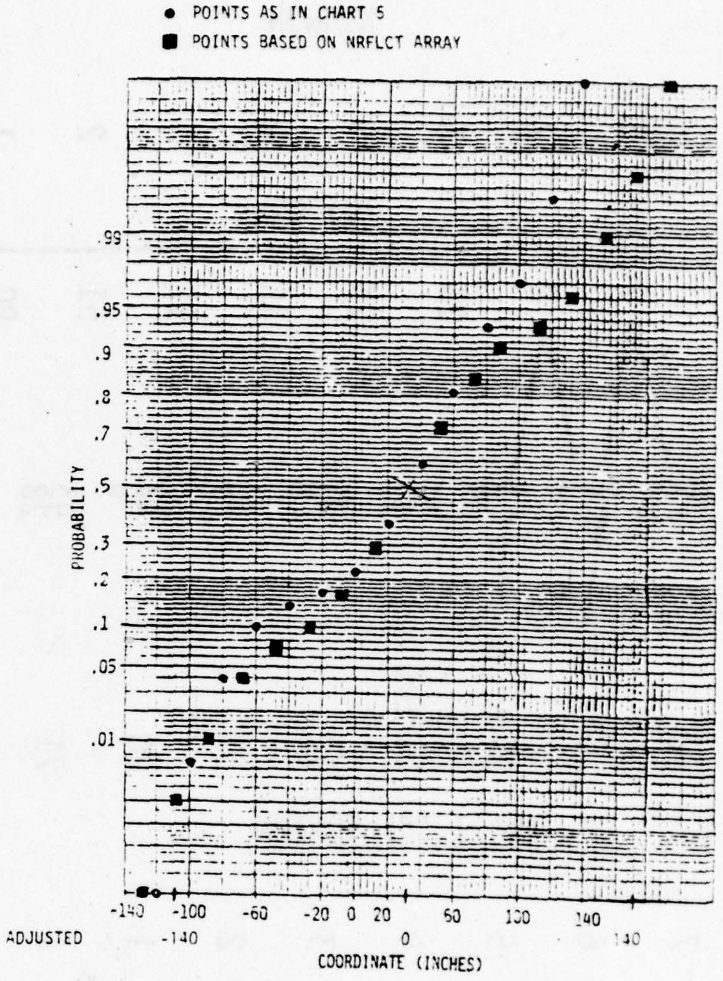


CHART 10 FRACTIONS BASED ON NRFLCT VALUES

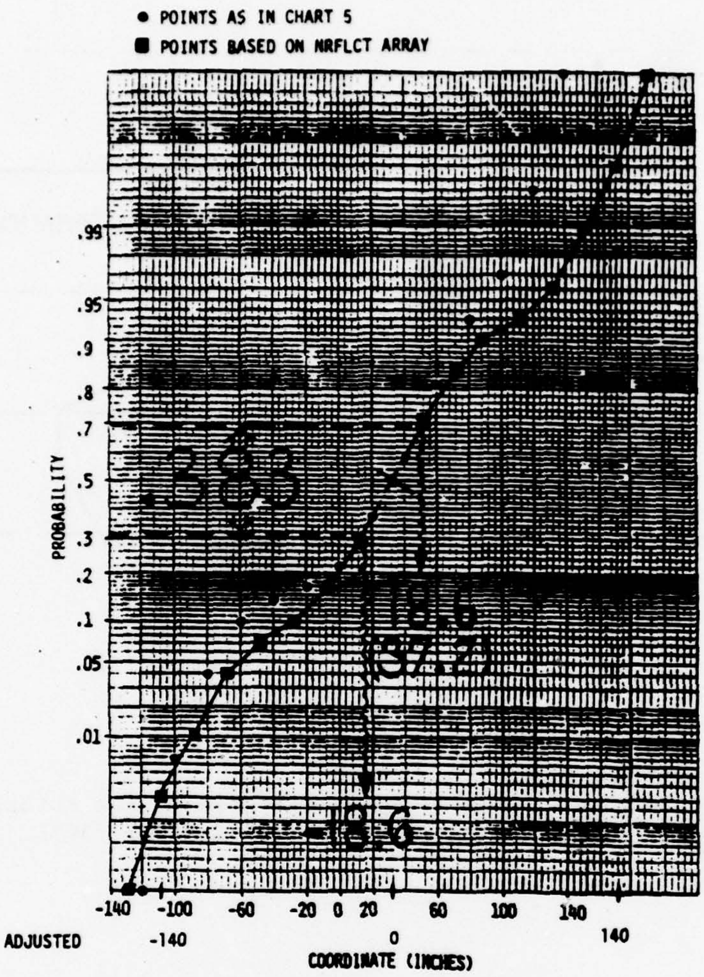


CHART 11 ESTIMATE OF 0.5 STANDARD DEVIATION

deviation. For example, as is shown in Chart 11, a frequency of .383 is associated with the mean plus or minus 0.5 standard deviation. Vertical lines through .5 - (.383/2), which equals about .31, and through .5 + (.383/2), approximately .69, cross line segments joining points based on the NRFLCT array at intersections that correspond to adjusted horizontal scale values of -18.6 and 18.6 inches. The related standard deviation estimate is 37.2 inches. Similarly, standard deviation estimates based on 1.0 and 1.5 times the standard deviation turn out to be 39.9 and 53.4 inches respectively.

3. CONCLUSION. Consider again the context within which arose the need for the procedure explained in this paper. Chart 12 illustrates the information concerning the vertical coordinates of missing first rounds that is conveyed by the calculated mean and a standard deviation estimate of 40 inches, selected as a best estimate for the particular situation used as an illustrative example in this paper.

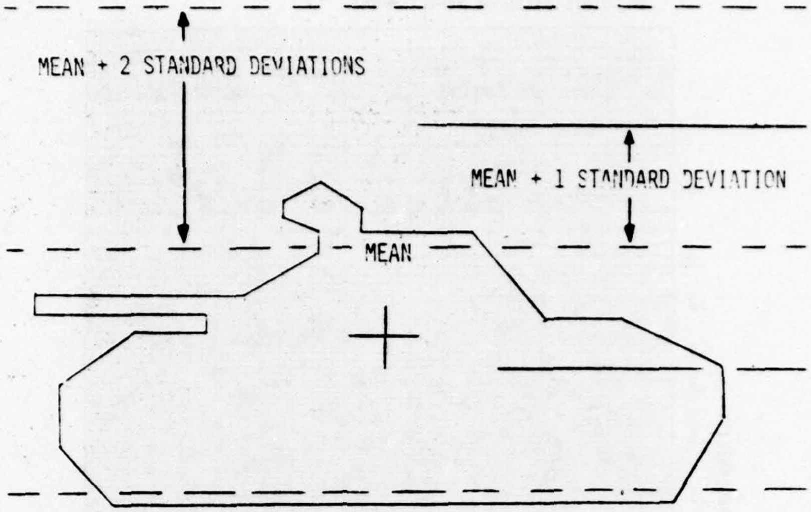


CHART 12 ILLUSTRATION OF DISTRIBUTION FOR MEAN OF 32 INCHES AND STANDARD DEVIATION OF 40 INCHES

Questions concerning this paper can be directed to the author in writing or by calling area code 301 278-4273/3675 or AUTOVON 283-4273/3675.

A PROCEDURE FOR CONSOLIDATING LINE OF SIGHT DATA

Bernard N. Goulet
U.S. Army Materiel Systems Analysis Activity
Aberdeen Proving Ground, Maryland 21005

ABSTRACT. Some large scale computer simulation models of ground combat involve computerized processing of detailed data representing terrain and tactical disposition of forces to determine line of sight relationships between attackers and defenders. These relationships are subject to change as the battle progresses because of such factors as maneuvering by vehicles or effects of obscuration. This paper describes a procedure for consolidating large amounts of line of sight data varying with time to help one better understand interactions between the nature of the simulated terrain, the choice of defender positions, and the maneuver plan of the attackers. Although this procedure has been applied manually to a single simulation situation only, computerization for large scale use would not be difficult.

1. INTRODUCTION. The U.S. Army uses several large scale computer simulation models of ground combat in support of weapon effectiveness studies. One of these is the AMSWAG model. AMSWAG is an acronym for AMSAA (Army Materiel Systems Analysis Activity) War Game. Chart 1 illustrates the general framework that is basic to AMSWAG and to certain other Army models. Important elements include the terrain representation, the selection of defender weapon positions, and the attacking force's plan of maneuver. Attacking maneuver weapons are assigned to one of up to three axes of advance and, within each axis, to one of several routes. Weapons on a route are further assignable to sections.

A common feature of many ground combat simulation models is that they involve computerized processing of detailed data representing terrain and tactical disposition of forces to determine line of sight relationships between attackers and defenders. In actual combat, these relationships are not fixed for the entire battle, but change to reflect such factors as the maneuvering of attacking vehicles or the effects of obscuration. AMSWAG includes a special preprocessor computer program, called LOSP1, which generates such line of sight information prior to the main AMSWAG runs. The LOSP1 program reflects the influence of terrain, vegetation, defender positions, and attacker maneuvering; obscuration effects are not presently included, however.

The basic concept underlying the AMSWAG line of sight preprocessor program is illustrated in Chart 2. The terrain profile between an observing defender and an attacking vehicle is broken into segments of approximately 25 meters each. Each segment is processed in turn to determine the extent of any screening of the attacker's vehicle. Vegetation height is superimposed on terrain profile elevation as applicable. Only the height of the attacking vehicle influences the output; horizontal dimensions of the vehicle are not considered for the line of sight calculations.

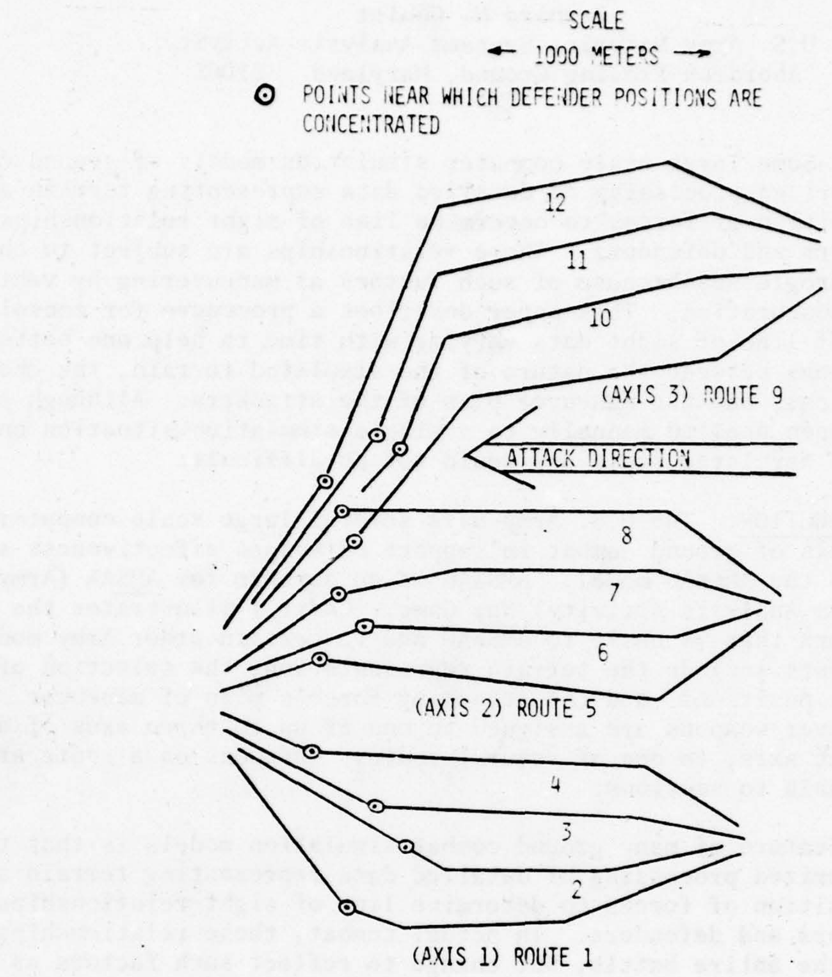


CHART 1 SAMPLE ATTACK PLAN

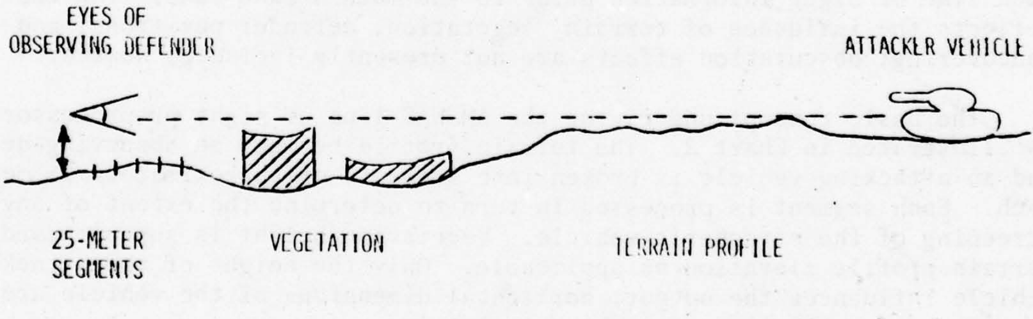


CHART 2 BASIC CONCEPT FOR DETERMINING LINE OF SIGHT RELATIONSHIPS

In connection with a recent AMSAA review of the exposure of armored vehicles in stationary defender positions, the AMSWAG LOSP1 program was modified to generate line of sight information somewhat different from that normally needed. Instead of data describing attacker exposure to a defender, the modified program provides detailed information concerning how much of a two meter high target located at each defender position would be exposed to each maneuvering attacker. The principal purpose of this paper is to describe how the line of sight data generated by the modified LOSP1 program were consolidated to facilitate understanding interactions between the nature of the simulated terrain, the choice of defender positions, and the maneuver plan of the attacking force. Although the consolidation procedure has been applied only manually to a single simulation situation, computerization for larger scale use would not be difficult.

2. BASIC DATA. Chart 3 illustrates a portion of the line of sight data generated by the modified LOSP1 program for defender weapon number 5. Such line of sight information is interpretable as a description of the degree of exposure of this weapon with reference to the attackers associated with various axes, routes, and sections. The data included in the chart have been selected so as to constitute a suitable basis for subsequent explanations. Lines of data each correspond to a particular time interval of 10 seconds, the basic interval used in the AMSWAG simulation. The various possible combinations of axis, route, and section are identified by the column headings 1 through 24. Even-numbered column headings do not appear because, to reduce the scope of calculations needed for this initial effort, only first sections were considered. Each numerical entry associated with line of sight is either zero or a four-digit number. Zeros are used for an entire column to identify possible attacker paths that were excluded from the calculations, again to limit the calculation scope. All four-digit numbers are to be interpreted as four separate single digits. The explanations that follow are based on reading of these digits from the left. If the lower quarter of the defender weapon is visible to an armored vehicle at the attacker location, the first digit is 1; otherwise, it is set to 9. If the next quarter moving up is visible, the second digit is set to 2; otherwise, it should be 9. Similarly, the third digit is 3 if the third quarter of the target is visible, and 9 if it is not. Finally, the fourth digit is 4 or 9 according to whether or not the top quarter is visible. The computer program methodology dictates that, whenever any quarter can be seen, portions of the target that are above that quarter are necessarily also visible. Incidentally, the chart illustrates the phenomenon that, for the later time intervals corresponding to relatively short distances between the attacking and defending forces, defenders tend not to be screened at all by intervening terrain and vegetation.

3. DESCRIPTION OF CONSOLIDATION PROCEDURE. The complete output data tabulation was voluminous and needed to be aggregated in some reasonable fashion in order to be useful. The steps involved in the aggregation process used are explained next.

a. Consider initially only defender weapon number 5. According to the scenario, this defender has the attacker weapons associated with axis 1 as its primary concern. Note that the first four columns of exposure data in Chart 3 correspond to axis 1. Axis 1 data were aggregated for groups consisting of an arbitrarily chosen 5 time intervals, such as 1950-2000 seconds. Observe that, for this 50-second group, attackers on axis 1, route 1, section 1 can see defender 5 during the time interval 1990-2000. Consequently, this defender is

```
DEFENDER    TIME          COMPUTER PROGRAM OUTPUT FOR VARIOUS INDICATORS OF
 WEAPON   INTERVAL          ATTACKER MANEUVER WEAPON AXIS, ROUTE, SECTION *
 NUMBER   (SECONDS)    1    3    5    7    9   11   13   15   17   19   21   23
 ------   ---------  ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ---- ----

   5        0-  10   9999 9999 9999    0 9999 9999 9999 9999    0 9999    0    0

           10-  20   9999 9999 9999    0 9999 9999 9999 9999    0 9999    0    0

                               E    T    C            .    .    .

         1950-1960   9999 9999 9999    0 9999 9999 9999 9999    0 9999    0    0

         1960-1970   9999 1234 9999    0 9999 9999 9999 9999    0 9999    0    0

         1970-1980   9999 1234 9999    0 9999 9999 9999 9999    0 9999    0    0

         1980-1990   9999 9234 9999    0 9999 9999 9999 9999    0 9999    0    0

         1990-2000   1234 9234 9999    0 9999 9999 9999 9999    0 9999    0    0

                               E    T    C            .    .    .

         2330-2340   1234 1234 1234    0 1234 1234 1234 1234    0 9999    0    0

         2340-2350   1234 1234 1234    0 1234 1234 1234 1234    0 9999    0    0

                               E    T    C            .    .    .

         2440-2450   1234 1234 1234    0 1234 1234 1234 1234    0 1234    0    0

   * INDICATORS CORRESPOND TO
      AXIS    ROUTE  SECTION
 1     1        1       1
 2                      2
 3              2       1
 4                      2
 (SIMILARLY FOR 6,8,10,...,24)
 5              3       1
 7              4       1
 9     2        1       1
11              2       1
13              3       1
15              4       1
17     3        1       1
19              2       1
21              3       1
23              4       1
```

CHART 3 SELECTED EXPOSURE DATA FOR DEFENDER NUMBER 5 visible to these attackers at least a portion of the time during the 50 seconds of concern. Similarly, determine that defender 5 is visible to attackers on axis 1, route 2, section 1 and on axis 1, route 3, section 1. Record 3 out of 3 as an indicator of at least occasional intervisibility between defender 5 and axis 1 attackers. Ignore completely axis 1, route 4, section 1 for which no calculations were made. Also ignore axis, route, section combinations associated with axis 1 but involving a section other than the first section on a route, for the same reason. Consequently, the maximum possible number of axis, route, section combinations of interest at this point is 3 rather than some higher number reflecting all route, section combinations possibly associated with axis 1. Next, an average exposure, given that at least occasional intervisibility exists, can be estimated. Consider the 15 four-digit numbers that correspond to axis 1 and to times between 1950 and 2000. Count the total number of target quarters visible each for a 10-second interval, considering that 9999 corresponds to none, 9994 to one, 9934 to two, 9234 to three, and 1234 to four. A total of 19 is thus obtained. This total could not have exceeded 60, the product of 3

(1) NUMBER OF AXIS, ROUTE, SECTION COMBINATIONS FOR WHICH INTERVISIBILITY EXISTS FOR AT LEAST ONE 10-SECOND INTERVAL *
(2) TOTAL TARGET QUARTERS VISIBLE FOR 10-SECOND INTERVAL **
(3) AVERAGE FRACTION OF DEFENDER EXPOSED DURING 50-SECOND INTERVAL, GIVEN AT LEAST OCCASIONAL INTERVISIBILITY

TIME INTERVAL (SECONDS)	(1)	(2)	(3)
0- 50	0		
50- 100	0		
ETC ...			
1900-1950	0		
1950-2000	3	19 (60)	0.32
2000-2050	2	22 (40)	0.55
2050-2100	0		
2100-2150	0		
2150-2200	1	1 (20)	0.05
2200-2250	3	16 (60)	0.27
2250-2300	3	30 (60)	0.50
2300-2350	3	58 (60)	0.97
2350-2400	3	60 (60)	1.00
2400-2450	3	60 (60)	1.00

* MAXIMUM POSSIBLE IS 3.
** MAXIMUM POSSIBLE, BASED ON EXCLUSION OF ANY AXIS, ROUTE, SECTION FOR WHICH INTERVISIBILITY DOES NOT EXIST AT LEAST FOR ONE 10-SECOND INTERVAL, IS INDICATED IN PARENTHESES.

CHART 4 AGGREGATED DATA FOR DEFENDER NUMBER 5, AXIS 1 (attacker axis, route, section combinations involving intervisibility at least some of the time) times 5 (10-second intervals) times 4 (quarters per defender target). The ratio of 19 to 60, which is 0.32, is the average fraction of defender 5 that is exposed to an attacker, given the existence of intervisibility as specifically defined here. Note that such a ratio reflects longer or shorter exposure times as well as larger or smaller exposed target sizes. The results of this first step for all 50-second intervals are shown in Chart 4.

b. The previous step concerns only attackers located on the same axis as defender number 5. Apply the same methodology for attackers associated with axis 2, which is adjacent to the axis of principal concern to defender 5.

c. Similar results are also needed for attackers on axis 3 which is remote, compared to either of the other two axes, for defender 5.

DEFENDER WEAPON NUMBER	DEFENDER AXIS		ATTACKER AXIS
5	1		1
			2
			3
EACH OF OTHER	1		1
DEFENDERS ASSIGNED			2
TO AXIS 1			3
EACH OF DEFENDERS	2		2
ASSIGNED TO			1, 3
AXIS 2			
EACH OF DEFENDERS	3		3
ASSIGNED TO			2
AXIS 3			1
ALL DEFENDERS	1	(A)	1
ASSIGNED TO		(B)	2
AXIS 1		(C)	3
ALL DEFENDERS	2	(D)	2
ASSIGNED TO		(E)	1, 3
AXIS 2			
ALL DEFENDERS	3	(F)	3
ASSIGNED TO		(G)	2
AXIS 3		(H)	1
COMBINED DATA FROM			
(A), (D), (F);			
(B), (E), (G);			
(C), (H)			

CHART 5 SEQUENCE OF STEPS FOR AGGREGATING DEFENDER EXPOSURE DATA

d. Repeat everything done so far for each of the other defender weapons assigned to axis 1.

e. Chart 5 lists the previous steps and shows the sequencing according to which the process is to be continued. The combination of data from (A), (D), and (F), illustrated in Chart 6, provides results applicable to all defenders with reference to attackers on the same axis. Similarly, by combining data from (B), (E), and (G) or from (C) and (H), one obtains overall results for all defenders with reference to attackers on an adjacent axis or on a remote axis.

It would be possible to combine data from all of (A) through (H) to get results for all defenders with reference to attackers on all axes. The degree of usefulness of such results is not clear, however, because defenders would not be fired at in combat equally frequently by attackers on a remote axis, on an adjacent axis, and on the same axis. If the aggregation scheme described here were used without any adjustment, equal frequency weightings would be effectively implied.

Application of the procedure just described yielded substantial quantitative information as a function of battle time. An additional step involved developing an approximate correlation between the battle time and the distance separating defender weapons and attackers. This correlation between times and distances was estimated from a graph on which attacker weapon positions at various times and defensive positions had been plotted. Determinations of separation distances were based on the frontal sectors of responsibility of the attacker and defender weapons considered. The graphical results that follow are plotted as a function of separation distance between attackers and defenders, which is judged a more directly useful parameter than simulation battle time. Note that the use of separation distance facilitates identification of data that may be associated with distances beyond probable weapon engagement ranges and thus of little or no concern.

4. CONCLUSION. Selected results are presented graphically in Charts 7 through 12. Each of these charts consists of upper and lower portions with a common horizontal scale, which is the distance between the attackers and the defenders. The vertical scale for the upper half of Charts 7 through 9 is the fraction of attackers that can see defender number 5, while the vertical scale for the lower half is the average exposure fraction for this defender given some intervisibility as defined throughout this paper. The attackers considered are on the same axis as defender number 5, on an adjacent axis, or on a remote axis as indicated in the title of each chart. The vertical scales in the two halves of Charts 10 through 12 are like those in the previous three charts, but are associated with consolidated information for all defenders as well as for the attackers identified by the chart titles.

An example can be included to indicate how the information contained more or less explicitly in the selected charts can be used to understand certain underlying relationships. Chart 7 indicates that defender number 5 is not exposed at all to attackers in its frontal sector until the attacking force closes to less than 1000 meters, but Chart 8 shows that this defender is not so totally hidden from attackers in the adjacent sector while the attacker is closing from about 2000 to 1000 meters. One can also see from Charts 10 and 11 that the exposure conditions applicable, to a specific position, such as that of defender number 5, may not always be representative of the exposure obtained by averaging for several defenders.

```
(1) NUMBER OF AXIS, ROUTE, SECTION COMBINATIONS FOR WHICH
     INTERVISIBILITY EXISTS FOR AT LEAST ONE 10-SECOND
     INTERVAL
(2) TOTAL TARGET QUARTERS VISIBLE FOR 10-SECOND INTERVAL **
(3) AVERAGE FRACTION OF DEFENDER EXPOSED DURING 50-SECOND
     INTERVAL, GIVEN AT LEAST OCCASIONAL INTERVISIBILITY

   TIME        AXIS 1         AXIS 2          AXIS 3               ALL
 INTERVAL    DEFENDERS      DEFENDERS       DEFENDERS            DEFENDERS
(SECONDS)    (1)  (2)       (1)  (2)        (1)  (2)         (1)    (2)       (3)
---------    ---  ---       ---  ---        ---  ---         ---    ---       ---

   0-  50     0              0               0               0

  50- 100     0              0               0               0

 ETC ...

 450- 500     0              0               0               0

 500- 550     0              0               2   8 ( 40)     2      8 ( 40)  0.20

 550- 600     0              4  32 ( 80)     2  40 ( 40)     6     72 (120)  0.60

 600- 650     0              4  80 ( 80)     2  40 ( 40)     6    120 (120)  1.00

 650- 700     0              4  80 ( 80)     2  40 ( 40)     6    120 (120)  1.00

 700- 750     3  36 ( 60)    4  80 ( 80)     2  40 ( 40)     9    156 (180)  0.67

 750- 800     3  60 ( 60)   11 121 (220)     2  40 ( 40)    16    221 (320)  0.70

 800- 850     3  60 ( 60)   16 274 (320)     2  40 ( 40)    21    374 (420)  0.90

 850- 900     3  60 ( 60)   16 320 (320)     2  40 ( 40)    21    420 (420)  1.00

 900- 950     3  60 ( 60)   16 320 (320)     2  40 ( 40)    21    420 (420)  1.00

 950-1000     3  60 ( 60)   16 320 (320)     2  40 ( 40)    21    420 (420)  1.00

1000-1050     5  68 (100)   16 320 (320)     2  40 ( 40)    23    428 (460)  0.93

1050-1100     3  60 ( 60)   16 320 (320)     2  40 ( 40)    21    420 (420)  1.00

1100-1150     3  60 ( 60)   16 320 (320)     2  40 ( 40)    21    420 (420)  1.00

1150-1200     3  60 ( 60)   16 320 (320)     2  40 ( 40)    21    420 (420)  1.00

1200-1250     3  58 ( 60)   16 320 (320)     2  40 ( 40)    21    418 (420)  1.00

1250-1300     2  40 ( 40)   16 320 (320)     2  40 ( 40)    20    400 (400)  1.00

1300-1350     2  29 ( 40)   16 320 (320)     2  40 ( 40)    20    389 (400)  0.97

1350-1400     1  14 ( 20)   16 320 (320)     2     ( 40)    19    374 (380)  0.98
```

CHART 6 DATA AGGREGATION FOR ALL DEFENDERS
WITH REFERENCE TO SAME AXIS

1400-1450	3	36	(60)	16	320	(320)	2		(40)	21	396	(420)	0.95
1450-1500	3	60	(60)	16	320	(320)	2		(40)	21	420	(420)	1.00
1500-1550	4	64	(80)	16	320	(320)	2		(40)	22	424	(440)	0.96
1550-1600	4	80	(80)	16	320	(320)	2		(40)	22	440	(440)	1.00
1600-1650	5	92	(100)	16	320	(320)	2		(40)	23	452	(460)	0.98
1650-1700	5	100	(100)	16	320	(320)	2		(40)	23	460	(460)	1.00
1700-1750	5	80	(100)	16	320	(320)	2		(40)	23	440	(460)	0.95
1750-1800	3	60	(60)	16	320	(320)	2		(40)	21	420	(420)	1.00
1800-1850	3	52	(60)	16	320	(320)	2		(40)	21	412	(420)	0.98
1850-1900	2	40	(40)	16	320	(320)	2	40	(40)	20	400	(400)	1.00
1900-1950	4	64	(80)	16	320	(320)	2	29	(40)	22	413	(440)	0.95
1950-2000	9	107	(180)	16	301	(320)	2	29	(40)	27	437	(540)	0.80
2000-2050	8	142	(160)	16	298	(320)	2	40	(40)	26	480	(520)	0.92
2050-2100	6	118	(120)	16	305	(320)	2		(40)	24	463	(480)	0.96
2100-2150	5	72	(100)	16	295	(320)	2		(40)	23	407	(460)	0.88
2150-2200	4	61	(80)	16	274	(320)	2		(40)	22	375	(440)	0.85
2200-2250	8	100	(160)	15	269	(300)	2		(40)	25	409	(500)	0.82
2250-2300	8	107	(160)	16	296	(320)	2		(40)	26	443	(520)	0.85
2300-2350	8	142	(160)	16	280	(320)	2	40	(40)	26	462	(520)	0.90
2350-2400	7	122	(140)	6	72	(120)	2	36	(40)	15	230	(300)	0.77
2400-2450	6	120	(120)	1	2	(20)	1	8	(20)	8	130	(160)	0.81

** MAXIMUM POSSIBLE, BASED ON EXCLUSION OF ANY AXIS,ROUTE, SECTION FOR WHICH INTERVISIBILITY DOES NOT EXIST AT LEAST FOR ONE 10-SECOND INTERVAL, IS INDICATED IN PARENTHESES.

CHART 6 (CONTINUED)

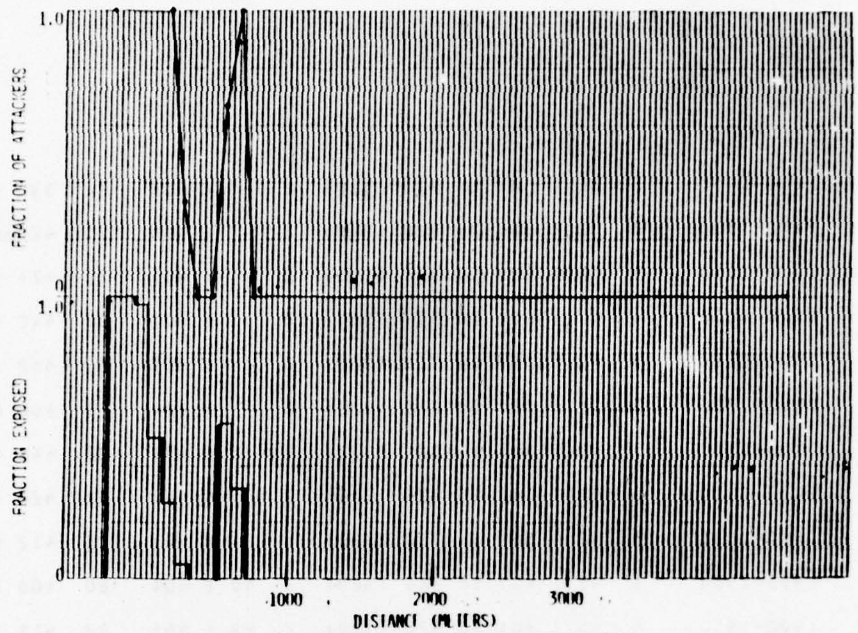


CHART 7 EXPOSURE DATA FOR DEFENDER 5, SAME AXIS

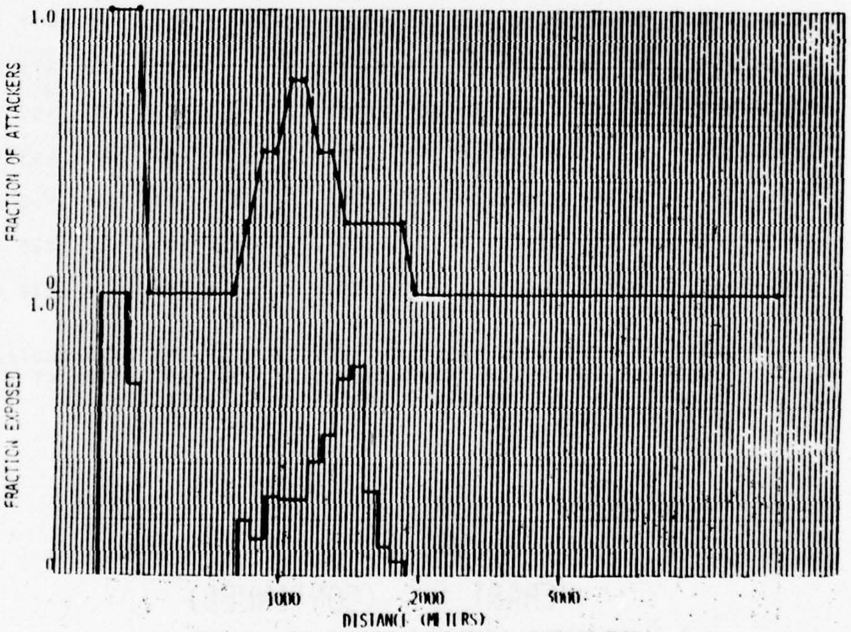


CHART 8 EXPOSURE DATA FOR DEFENDER 5, ADJACENT AXIS

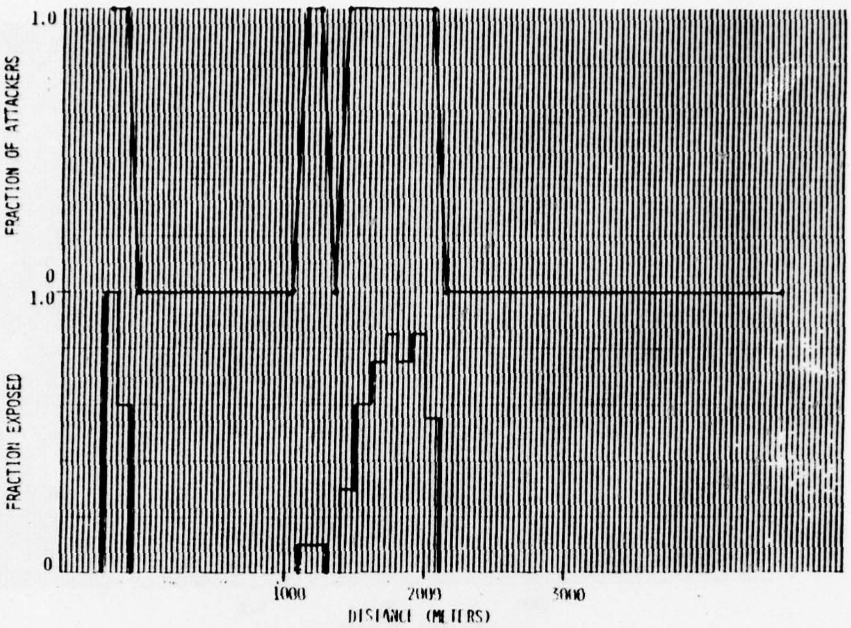


CHART 9 EXPOSURE DATA FOR DEFENDER 5, REMOTE AXIS

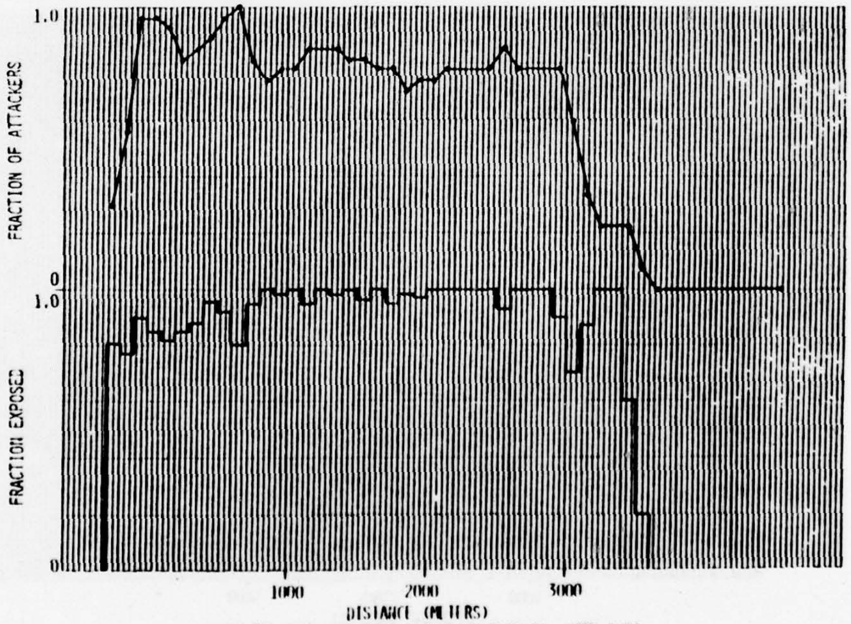


CHART 10 EXPOSURE DATA FOR ALL DEFENDERS, SAME AXIS

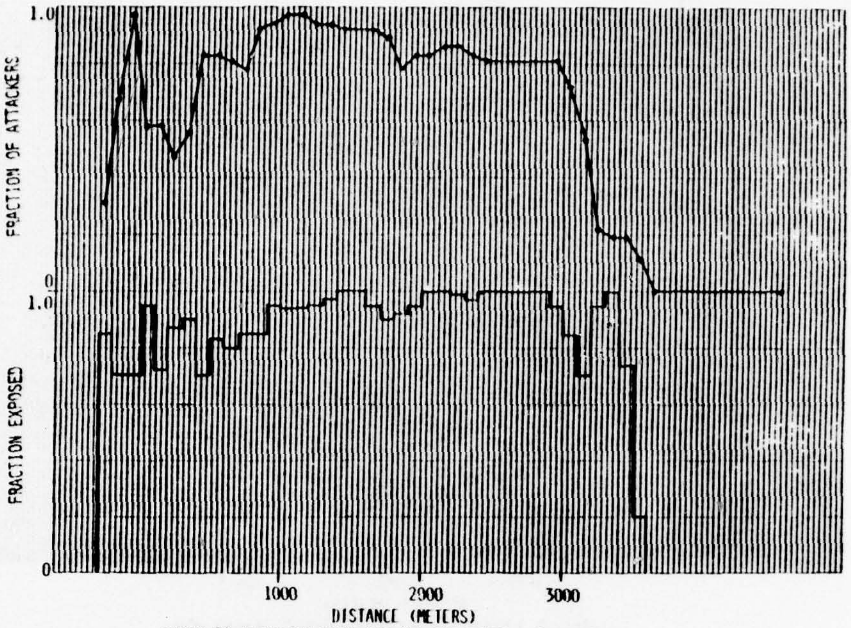


CHART 11 EXPOSURE DATA FOR ALL DEFENDERS, ADJACENT AXIS

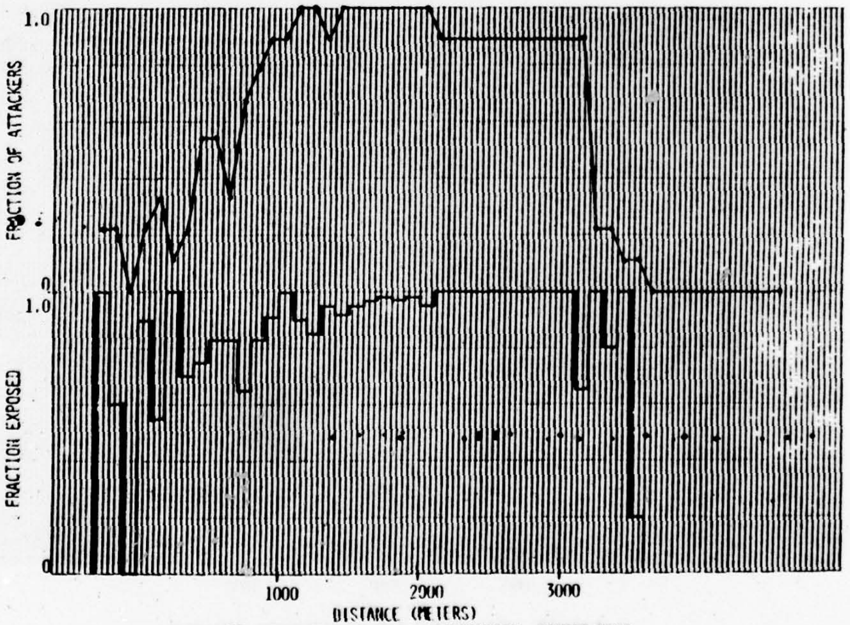


CHART 12 EXPOSURE DATA FOR ALL DEFENDERS, REMOTE AXIS

Finally, it may be useful to emphasize the consolidation procedure explained in this paper involves a sort of averaging over time and exposed target size as well as over various attackers and eventually over various defenders. The derived data are equivalent only approximately, rather than rigorously, to the original data. One must sacrifice some mathematical rigor in order to reduce huge amounts of detailed numerical information to a form that more readily permits its assimilation and use by a human mind.

Questions concerning this paper can be directed to the author in writing or by calling area code 301 278-4273/3675 or AUTOVON 283-4273/3675.

ROBUST SOFTWARE FOR ROBUST STATISTICS

Virginia C. Klema
Massachusetts Institute of Technology

ABSTRACT

Iteratively reweighted least squares is a part of robust statistics where "robustness" means relative insensitivity to moderate departures from statistical assumptions. Software is "robust" if it is reliable and its performance is not degraded by the computer or operating-system environment in which it is used. ROSEPACK (RObust Statistics Estimation PACKage) is a system of portable Fortran subroutines and an interactive driver cast as robust software to do the iteratively reweighted least squares problem. We describe the software environment in which ROSEPACK was constructed and how it can be used.

INTRODUCTION

We address the problem of providing reliable and portable software for iteratively reweighted least squares which is a part of robust statistics. This work on mathematical and statistical software represents an interdisciplinary mix of data analysis, numerical analysis, and software engineering, three areas in which rapid progress has been made in recent years. Software that is reliable, portable, and well-documented provides a vehicle to stimulate research in data analysis and numerical analysis. ROSEPACK (RObust Statistics Estimation PACKage) is such a vehicle.

The work on ROSEPACK represents cooperation and collaboration among data analysts and numerical analysts. Paul Holland designed the algorithms for iteratively reweighted least squares. Roy Welsch collaborated with Paul Holland to provide the default tuning constants for the eight weight functions in ROSEPACK. David Hoaglin designed the software for the stem-and-leaf-display, and Stanley Wasserman helped in programming and testing the software for stem-and-leaf in ROSEPACK. Hoaglin, Holland, and Welsch patiently and repatedly explained much of robust statistics to us. Richard Bartels and Andrew Conn shared their work and code on the overdetermined solution in the ℓ_1 norm and were kind enough to use some of the modules in ROSEPACK that assure portability that might otherwise not obtain for generation of pseudo-random numbers.

Concurrent work on design and implementation of algorithms for nonlinear least squares, NL2SOL [6], was of much benefit to ROSEPACK. John Dennis suggested the numerical convergence criterion used in the iterations. David Gay programmed the condition estimator and helped test the pseudo-random numbers used for the right hand side by the condition estimator. The estimator itself repreents the work of Alan Cline, Cleve Moler, G.W. Stewart, and J.H. Wilkinson. The use of the estimator in ROSEPACK makes possible the efficient monitoring of the condition of the least squares problem at each iteration.

Neil Kaden, David Coleman, and Stephen Peters worked as programming and research assistants on the project. Scott Matthews independently checked the documentation for using the interactive driver and the subroutines. Paul Velleman used the software heavily in his own research and gave us several useful comments, many of which we incorporated in the interactive driver.

We gratefully acknowledge the support of the National Science Foundation for this research*.

*National Science Foundation Grants #DCR 75-08802, MCS 76-11989, and MCS 77-12514.

SECTION 1

Robustness

The word "robust" is defined in the Random House College Dictionary as

1) strong and healthy, hardy, vigorous;
2) strongly or stoutly built
3) suited to or requiring bodily strength or endurance;
4) rough, rude, or boisterous;
5) rich and full bodied;

and has synonyms, "powerful, sound" for the first definition and "course, rambunctious" for the fourth definition. We are persuaded that the first rather than the second set of synonyms corresponds to the terms "robust statistics" and "robust software."

Robustness has a common thread of meaning in statistics and mathematical software although certainly its use in statistics preceded its use in mathematical software by more than a decade. For statistical purposes Peter Huber [10] writes "robustness signifies insensitivity against small deviations from the assumptions." Huber [10] states further that for most practical purposes "distributionally robust" and "outlier resistance" are interchangeable.

"Robustness" of mathematical software is defined by Cody [4] to mean the ability of a program to detect and gracefully recover from abnormal situations without terminating the computer run. Robust mathematical software is reliable in the sense that its performance is unaffected by the computer or operating system environment in which it is run. Such reliability occurs if the software contains sufficient safeguards against arithmetic exceptions and is written to be portable across machine lines. Robust mathematical software continues computation as long as meaningful results can be obtained, and if such computation cannot continue clear indication in the form of error recovery must show where and why the algorithm or the software is failing.

Portability of mathematical software is required if one expects the software to be executed correctly on more than one computer or in more than one computing environment. The computing environment includes the compiler, the operating system, and, at times, applications subsystems or statistical software packages.

Portability of software is achieved primarily through the use of disciplined Fortran best described by Brian Smith [16]. Portability is assured by careful testing, consistent documentation, and wide use. Much attention has been given to requirements for constructing portable software. We do not address these requirements in this paper but refer the reader to the carefully written book, "Portability of Numerical Software," Lecture Notes in Computer Science 57, edited by Wayne Cowell, Springer-Verlag, 1977.

SECTION 2

Iteratively Reweighted Least Squares

Regression, which assumes certain statistical hypotheses, or ordinary least squares, which need not assume such hypotheses, has been widely used for fitting models to data. It is versatile and the algorithms to compute the least squares solutions are well understood. Despite its central role in the past much work has been done by statisticians in recent years to strengthen least squares and gain more information than is generally available about the data in the sense that outliers or leverage points may influence fitting data to the model. One goal of robust regression is to avoid undue influence on the fit if there are slight changes to all of the data or large changes to a few of the data points. The problem of iteratively reweighted least squares (IRLS) is one way of pursuing this goal for robust linear regression.

Consider the statistical model

$$\underset{\sim}{y} = X\underset{\sim}{\beta} + \underset{\sim}{e}$$

where y is an n vector of observations corresponding to the rows of X, X is an nxp design matrix or data matrix of n observations and p parameters, e is an n vector about which various statistical assumptions are made, and β is a p vector.

The ordinary least squares problem is

$$\min_{\beta} \sum_{i=1}^{n} ((r_i(\beta))/s)^2$$

where r is a vector of residuals $\underset{\sim}{y}-X\hat{\underset{\sim}{\beta}}$, and s is a constant or fixed scale.

The weighted least squares problem is

$$\min_{\beta} \sum_{i=1}^{n} W_i((r_i(\beta))/s)^2$$

where W is a diagonal matrix of weights that are functions of scaled residuals.

The iteratively reweighted least squares problem assumes a start $\hat{\beta}_0$ which can be obtained from ordinary least squares, ℓ_2, least absolute residuals, ℓ_1, previous iterations of IRLS, or a start specified by the user. Given $\hat{\beta}_0$, IRLS gives

$$\hat{\underset{\approx}{\beta}}^{(k+1)} = \left(\left(W^{(k+1)}\right)^{1/2} X^{+}\right) \left(W^{(k+1)}\right)^{1/2} \underset{\sim}{y}$$

where X^+ represents the pseudo-inverse of X.

The start, $\hat{\underset{\sim}{\beta}}_0$, can be computed as an ℓ_2 start from ordinary least squares, or an ℓ_1 start from the overdetermined solution in the ℓ_1 norm. The ℓ_2 start is computed by Householder transformations, QR with column pivoting, [18] or a combination of QR and the singular value decomposition, MINFIT [7]. Precautions are taken in the versions of QR and MINFIT in ROSEPACK to avoid arithmetic exceptions. The ℓ_1 start is computed by CL1 [2].

The problem of IRLS is an optimization problem in the sense of minimizing a function of scaled residuals. The residual scaling function used in ROSEPACK is the median absolute deviation, i.e., the median of the absolute value of nonzero residuals. The inclusion of additional residual scaling functions such as the inter-quartile-range is readily possible.

The convergence criterion for IRLS suggested by John Dennis [5] is a scale-independent measure of the gradient, $X^T\underset{\sim}{r}$ where $\underset{\sim}{r}$ is the residuals $\underset{\sim}{y}-X\hat{\underset{\sim}{\beta}}$. After the k^{th} iteration convergence is checked by computing.

$$\left(\left(\left(W^{(k+1)}\right)^{1/2}X_j\right)^T\left(\left(W^{(k+1)}\right)^{1/2}\underset{\sim}{r}^{(k)}\right)\right) \Big/$$

$$\left\|\left(W^{(k+1)\,1/2}\right)X_j\right\|_2 \left\|\left(W^{(k+1)}\right)^{1/2}\underset{\sim}{r}^{(k)}\right\|_2 \quad .$$

The function that is being minimized determines the weighting function that is used and thus the elements of the diagonal matrix W. We minimize

$$\sum_{i=1}^{n}\rho(r_i(\beta))/s$$

and

$$W(u) = \frac{\rho'(u)}{|u|} \quad .$$

ROSEPACK contains the eight wright functions given in Table 1.

We do not consider further details here on IRLS but refer the reader to Holland and Welsch, 1977 [11].

The computational algorithms to compute the ℓ_2 start and iterations subsequent to any start are orthogonal factorizations that are Householder transformations, QR, with column pivoting, or the singular value decomposition, MINFIT [7] or a combination of these. By option a user may select the ℓ_1 start or ℓ_2 with singular value decomposition, however ROSEPACK normally defaults to QR for ℓ_2. Throughout the ℓ_2 start and iterations of IRLS numerical rank [9] must be estimated or determined. Unless X is exactly singular the QR factorization gives the upper triangular matrix R whose linear condition is that of X. The condition estimator that we use is a variant of the estimator [3]. Our variation of the estimator uses a machine independent pseudo-random vector for the right hand side.

The condition estimate is computed in $0(p^2)$ operations for an n by p matrix and is an estimate of the largest and the smallest singular values, σ_{max} and σ_{min}, of X.

If the estimated ratio $(\sigma_{min}/\sigma_{max})$ is less than the certainty of the data as specified by the user or $\varepsilon^{1/2}$ where ε is the relative precision of the arithmetic of the computing machine, computation continues with explicit rank determination by the singular value decomposition.

We provide column equilibration as an option in ROSEPACK. However, we strongly urge potential users to select column equilibration (or scaling) in light of what they believe about the certainty of their data.

SECTION 3

The Environment In Which ROSEPACK Was Constructed

The work on ROSEPACK was started in May, 1975 at the Computer Research Center of the National Bureau of Economic Research and was supported by the National Science Foundation. The research was an interdisciplinary mix of data analysis, numerical analysis, and software engineering. David Hoaglin, Paul Holland, and Roy Welsch defined the iteratively reweighted least squares problem. Gene Golub and John Dennis helped us design the numerical algorithms and the convergence criterion for the numerical calculations.

The host machine on which the software was constructed was an IBM 370/168 with the VM operating system under which we used CMS.

The subroutines and the interactive driver in ROSEPACK are necessarily written in FORTRAN. The work on the software and the documentation was strongly influenced by experience with EISPACK [17]. This experience revealed the requirement that the software and the documentation for robust software must be consistently disciplined. Nonetheless decisions about the design of a large software package are frequently arbitrary.

To assure consistent discipline in writing the software and its documentation we wrote a set of guidelines [12] that was revised and extended. These guidelines were criticized and then followed by the staff members and students who wrote ROSEPACK. Since the work was done over a period of three years a succession of undergraduate and graduate students participated in writing and testing the subroutines and interactive driver.

Our goal in the preliminary design of ROSEPACK was to use existing robust software wherever possible and to write the new software in modular subroutine form. Each module contains sufficient documentation for use and can be used independently or as a part of the package with the interactive driver. We used adjustable dimensions for storage of arrays, and we include documentation that shows a user how to modify storage if he needs to do so for his particular problem.

We wanted ROSEPACK to be operational on non-IBM computing machines without manual modification of the code or its documentation. Furthermore, we wanted to avoid interrogating the computing machine in order to determine underflow, overflow, and precision limits. To meet these requirements and to make subsequent distribution straightforward we designed the software so that the IMSL converter [1] inserts the appropriate machine-dependent constants. All variables and intrinsic functions are declared, a feature that allows the IMSL Converter to produce either a long or a short precision version of the code for a particular computer.

To check flow of control and considerations for portability we used the PFORT verifier [15] to check each subroutine and, finally, the execution of the collection of subroutines and the interactive driver.

The above decisions had to do mostly with processing, verifying, and distributing the code. Other decisions simply influenced the Fortran construction, discipline, and style of the code. For reasons of efficiency and conservation of execution time we expected the code to be compiled with the highest level of optimization. Therefore we did not optimize in the sense of linearizing addresses of elements of two-dimensional arrays. We did not use the BLAS [13,14]. Our reasons for not using the BLAS were three-fold. The efficiency of the BLAS was uncertain with respect to the construction of their inner loops. Different precisions of the BLAS would be required for our use. We did not want the overhead of extra subroutine calls to do the basic linear algebra.

ROSEPACK was tested at Hampshire College, Bell Laboratories, Educational Testing Service, Tufts University, and the University of California, San Diego before we thought it should be generally distributed or submitted for publication to Transactions on Mathematical Software.

SECTION 4

How ROSEPACK Can Be Used

ROSEPACK contains 47 subroutines, internal documentation for use, and an interactive driver that contains on-line documentation in the form of prompts and help-commands to assist the user. The interactive driver controls options for using an ℓ_1 or an ℓ_2 start for the iterations, selecting user-supplied weights or any of the eight weighting functions.

Input to ROSEPACK is in subroutine form and can be readily modified to handle file input. Output is initiated by a print control vector and is governed by the interactive driver. The (optional) output information includes the display, the number of observations, the number of variables, the maximum diagonal or all diagonal elements of the "hat" matrix [8] (QQ^T from QR or $U\Sigma\Sigma^+U^T$ from MINFIT) the condition number (or estimate) of the (weighted) data matrix, the weights, the minimum absolute residual, and the minimum weight. There is an option to print the (weighted) sum of squared residuals and the sum of absolute residuals. The R-squared statistic, the (weighted) standard error, and the (weighted) F statistic are available. All of the subroutines can be used in a batch or an interactive mode of computing. Output can be in the form of a stem-and-leaf display [19].

The interactive driver, the subroutines, a collection of test matrices, and a user's memo are included on a magnetic tape. Since January, 1978 ROSEPACK has been distributed by IMSL, Inc., Sixth Floor, GNB Bldg., 7500 Bellaire, Houston, Texas 77036. The charge for each tape distributed is $100.

As a consequence of the distribution of ROSEBACK by IMSL we have received some comments for users. Though we did not have the minicomputers particularly in mind when we wrote ROSEPACK, apparently the software is operational on at least one minicomputer - the PRIME. From the transfer to the PRIME we learned that we have one error in documentation. From a user on a Honeywell computer we found an error in our computation of the R-squared statistic. This error involves nine lines of code. Naturally we corrected the documentation and the coding mistake. We welcome the fact that users let us known about these problems.

A manuscript describing the algorithms in ROSEPACK and the software was submitted for publication in Transactions on Mathematical Software. The paper, "A System of Subroutines for Iteratively Reweighted Least Squares," by David Coleman, Paul Holland, Neil Kaden, Virginia Klema, and Stephen Peters will appear in a forthcoming issue of TOMS.

TABLE 1

Weight functions (where u = scaled residual), range and default tuning constants.

Name	w(u)	Range	Turning Constant*
ANDREWS	$w_A(u) = \begin{cases} \sin(u/A)/(u/A) \\ 0 \end{cases}$	$\lvert u\rvert < \pi A$ $\lvert u\rvert > \pi A$	A = 1.339
BIWEIGHT	$w_B(u) = \begin{cases} [1 - (u/B)^2]^2 \\ 0 \end{cases}$	$\lvert u\rvert \leq B$ $\lvert u\rvert > B$	B = 4.685
CAUCHY	$w_C(u) = 1/(1 + (u/C)^2)$		C = 2.385
FAIR	$w_F(u) = 1/(1 + \lvert u/F\rvert)$		F = 1.400
HUBER	$w_H(u) = \begin{cases} 1 \\ H/\lvert u\rvert \end{cases}$	$\lvert u\rvert \leq H$ $\lvert u\rvert > H$	H = 1.345
LOGISTIC	$w_L(u) = \tanh(u/L)/(u/L)$		L = 1.205
TALWAR	$w_T(u) = \begin{cases} 1 \\ 0 \end{cases}$	$\lvert u\rvert \leq T$ $\lvert u\rvert > T$	T = 2.795
WELSCH	$w_R(u) = e^{-(u/R)^2}$		R = 2.985

*These are default values for the tuning constants, which are designed to have 95% asymptotic efficiency with respect to ordinary least squares when the disturbances come from the normal or Gaussian distribution and the scaling function converges to the standard deviation of that disturbance distribution.

REFERENCES

1. Aird, T.J., "The Fortran Converter User's Guide," IMSL, (1975).

2. Bartels, R., and Conn, A., "Linearly Constrained ℓ_1 Problems," Johns Hopkins University TR 248, June (1976), to appear in Transactions on Mathematical Software.

3. Cline, A., Moler, C.B., Stewart, G.W., and Wilkinson, J.H., "On an Estimate for the Condition Number of a Matrix," to appear, Numerische Mathematik.

4. Cody, W.J., "The Construction of Numerical Subroutine Libraries," SIAM Review, Vol. 16, no. 1, pp. 36-46, January (1974).

5. Dennis, J.E., Jr., "Nonlinear Least Squares and Equations," The State of the Art in Numerical Analysis, Academic Press, (1977).

6. Dennis, J.E., Jr., Gay, David M., and Welsch, Roy E., "An Adaptive Nonlinear Least Squares Algorithm," National Bureau of Economic Research, Inc., W.P. 196, (1978).

7. Garbow, B.S., Boyle, J.M., Dongarra, J.J., and Moler, C.B., Matrix Eigensystem Routines - EISPACK Guide Extension, Lecture Notes in Computer Science 51, Springer-Verlag, (1977).

8. Hoaglin, David C., and Welsch, Roy E., "The Hat Matrix in Regression and ANOVA," the American Statistician, Vol. 32, no. 1, pp. 17-22, (1978).

9. Golub, G., Klema, V., and Stewart, G.W., "Rank Degeneracy and Least Squares Problems," University of Maryland TR-456, (1976), Stanford University, STAN-C5-76-559, (1976), National Bureau of Economic Research, Inc., WP 165, (1977).

10. Huber, Peter J., Robust Statistical Procedures, SIAM, (1977).

11. Holland, Paul W., and Welsch, Roy E., "Robust Regression Using Iteratively Reweighted Least Squares," Commun. Statist. - Theor. Meth. A6(9), pp. 813-827, (1977).

12. Kaden, N., and Klema, V., "Guidelines for Writing Semi-portable Fortran," National Bureau of Economic Research, Inc., WP 130, (1976).

13. Lawson, C.L., Hanson, R.J., Kincaid, D.R., and Krogh, F.T., "Basic Linear Algebra Subprograms for Fortran Usage," Report SAND 77-0898, Sandia Labs., (1977).

14. Lawson, C.L., Hanson, R.J., Kincaid, D.R., and Krogh, F.J., "Basic Linear Algebra Subprograms for Fortran Usage," Report CNA-124/TR-72, Center for Numerical Analysis, U. of Texas at Austin, (1977)

15. Ryder, B.G., "The PFORT Verifier: User's Guide," Computing Science Tech. Report 12, Bell Telephone Labs., (1975).

16. Smith, Brian, "Fortran Poisoning and Antidotes," in Portability of Numerical Software, Lecture Notes in Computer Science 57, Springer-Verlag, (1977).

17. Smith, B.T., Boyle, J.N., Dongarra, J.J., Garbou, B.S., Tkebe, Y., Klema, V.C., and Moler, C.B., Matrix Eigensystem Routines - EISPACK Guide, Second Edition, Lecture Notes in Computer Science 6, Springer-Verlag, (1976).

18. Stewart, G.W., Introduction to Matrix Computations, Academic Press (1973)

19. Tukey, John W., Exploratory Data Analysis, Addison Wesley, (1977).

ADJUSTMENT BY LEAST SQUARES IN GEODESY USING BLOCK ITERATIVE METHODS FOR SPARSE MATRICES

R. J. Plemmons*

Departments of Computer Science and Mathematics
The University of Tennessee
Knoxville, TN 37916

ABSTRACT

In this paper, 3-block iterative methods are applied to the least squares solution to large sparse overdetermined systems of linear equations associated with angle adjustment in geodetic position network problems. An SOR type method is shown to be effective in the case where weighting the equations is acceptable. For the general case the Chebyshev semi-iterative method applied to the Gauss-Seidel iterations is recommended. Alternative approaches such as the use of the normal equations or orthogonal decompositions are difficult to use on these adjustments due to problems with ill-conditioning, fill-in and storage. The methods given here are very simple to implement and use, and are especially attractive when storage and execution time are important considerations. Some successful numerical experiments are reported and possible applications of the methods to coordinate adjustments for triangulation systems associated with large amounts of satellite photographic data are discussed.

*This research was supported by the U. S. Army Research Office under grant No. DAAG29-77-G-0166.

1. Introduction. Geodetical network problems in the practice of surveying give rise to special kinds of very sparse overdetermined systems of linear equations. The current use by almost all surveyors of electronic distance measuring equipment and one-second theodolites for angle measurements necessitates modern adjustment procedures to guard against the possibility of blunders and to obtain a better estimate of the unknown quantities being measured. The number of observations is always larger than the minimum required to determine the unknowns. The relationships between the unknown quantities lead to an overdetermined system of linear equations. The measurements are then adjusted in the sense of least squares by computing a (weighted) least squares solution to this system, the values of this solution then giving the adjusted measurements.

The use of least squares methods for geodetical adjustments dates back at least to Gauss, who used the method of least squares as early as 1794 in determining the orbital elements of the earth and other planets. For an informative review of the historical background of least squares applications to geodesy made by scientists in the 19th and early 20th centuries see Veress [1974, Section 1.3]. Since 1950 an overall least squares adjustment approach has developed and its extended use in geodetical calculations has been made possible by the use of modern computers and the development of matrix methods in numerical analysis.

In general, a geodetical position network is a mathematical model consisting of several mesh-points or geodetic stations, with unknown positions over a reference surface or in three-dimensional space. These stations are connected by lines, each representing one or more observations involving the two stations terminating one line. The observations may be angles or distances; however, this paper will be primarily concerned with least squares adjustments of angle measurements. Figure 1 below illustrates a five-station geodetical network where at every station angles between neighboring stations are observed counterclockwise.

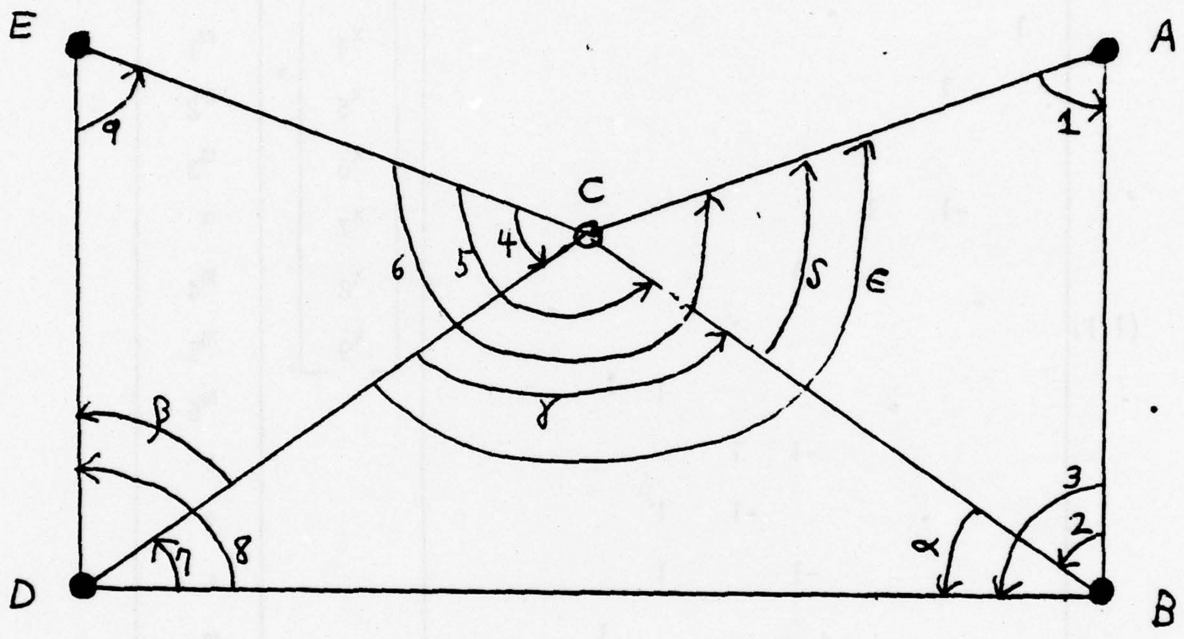


Figure 1

The unknown angles in the network are observed (measured) by the surveyor and the natural geometric relationships between the angles in the triangles are used to form an overdetermined system of linear equations relating these observations. Let

x_j = unknown angle j (in degrees), $1 \leq j \leq 9$.

The objective here is to adjust the measurements of these nine primary angles.

There results then an overdetermined system of linear equations of the form

$$(1.1)\quad \begin{bmatrix} 1 & 1 & & & -1 & 1 & & & \\ & -1 & 1 & -1 & 1 & & 1 & & \\ & & & 1 & & & -1 & 1 & 1 \\ 1 & & & & & & & & \\ & 1 & & & & & & & \\ & & 1 & & & & & & \\ & -1 & 1 & & & & & & \\ & & & 1 & & & & & \\ & & & & 1 & & & & \\ & & & & & 1 & & & \\ & & & -1 & 1 & & & & \\ & & & & -1 & 1 & & & \\ & & & -1 & & 1 & & & \\ & & & & & & 1 & & \\ & & & & & & & 1 & \\ & & & & & & -1 & 1 & \\ & & & & & & & & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{bmatrix} = \begin{bmatrix} 180^\circ \\ 180^\circ \\ 180^\circ \\ m_1 \\ m_2 \\ m_3 \\ \alpha \\ m_4 \\ m_5 \\ m_6 \\ \gamma \\ \delta \\ \varepsilon \\ m_7 \\ m_8 \\ \beta \\ m_9 \end{bmatrix}$$

Here, for example, the three individual triangle 180° sums become part of the system, but these may not be exact due, for example, to the curvature of the earth. Also, five of the observed angles, α through ε, are expressed as differences between primary angles x_j.

A second illustration of a geodedical network, this one with eight stations, is given below in Figure 2. Once again, all angles are measured counterclockwise and the primary angles between adjacent stations are numbered.

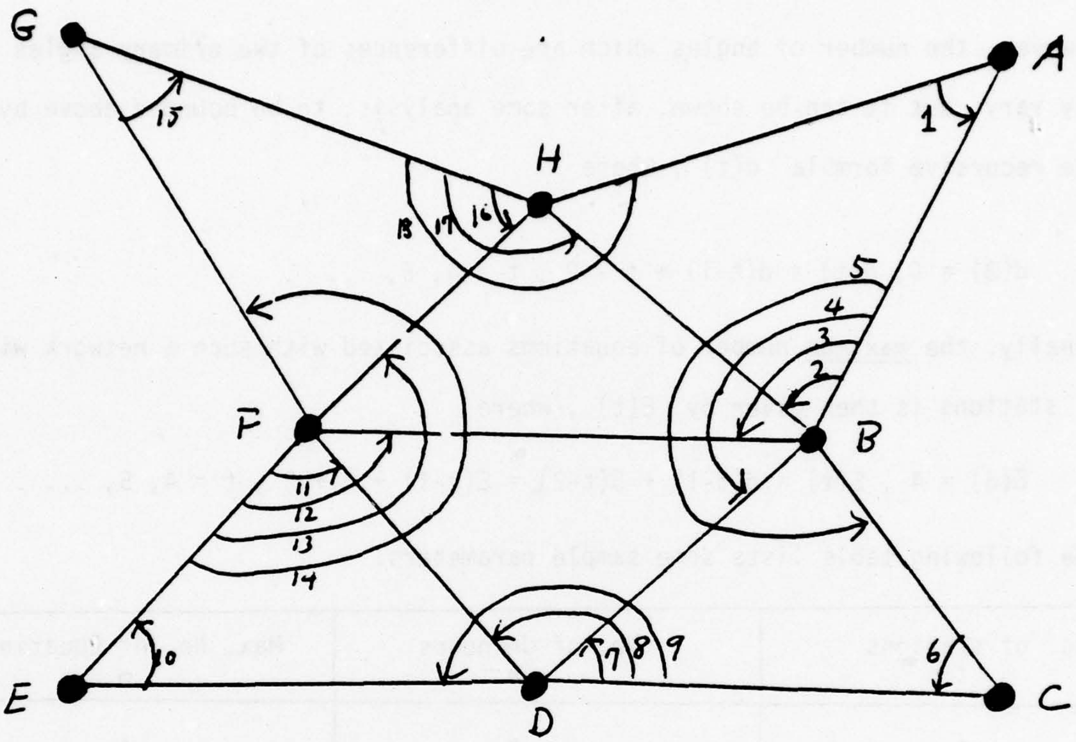


Figure 2

Note here that there are n = 18 primary angles, that there are six triangles and thus six equations involving 180° measurements, and there are 18 equations expressing measured angles as the difference between two primary angles. Thus the resulting overdetermined system of linear equations has n = 18 unknowns (measurements to be adjusted) and m = 18 + 6 + 18 = 42 equations.

If it is agreed that for the networks in question, the addition of one station results in the addition of one triangle to the network, then the parameters for the associated system of linear equations can be given as follows. Let

t = no. of stations in the network, t $\geq$ 3.

Then such a network always has $t - 2$ triangles and thus $t - 2$, 180° angle sum equations and exactly

$$n = 3(t-2) \text{ primary angles (unknowns).}$$

However, the number of angles which are differences of two primary angles may vary; but it can be shown, after some analysis, to be bounded above by the recursive formula $d(t)$, where

$$d(3) = 0,\ d(t) = d(t-1) + t - 2\ ,\ t = 4, 5, \ldots\ .$$

Finally, the maximum number of equations associated with such a network with t stations is then given by $E(t)$, where

$$E(3) = 4\ ,\ E(t) = d(t-1) + 5(t-2) = E(t-1) + t + 2\ ,\ t = 4, 5, \ldots\ .$$

The following table lists some sample parameters.

No. of stations t	No. of Unknowns n	Max. No. of Equations m
3	3	4
4	6	10
5	9	17
8	18	44
50	144	1,264
100	294	5,039

In general then, geodetical networks involving a large number of stations may lead to very large sparse overdetermined systems of linear equations of the form

$$(1.2) \qquad Ax = b$$

where A is $m \times n$ and $m \gg n$. Since a measurement is taken for each of the n primary angles, A has full column rank n. The least squares solution to (1.2) is then the unique solution to the problem: $\min_x \|b - Ax\|_2$

where $|| \; ||_2$ denotes the Euclidean norm. An equivalent formulation of the problem is the following: one seeks to determine vectors y and r such that

$$r + Ay = b$$
$$A^t r = 0.$$

It follows that the least squares solution to (1.1), where A has full column rank, is the unique solution to the nonsingular system of <u>normal equations</u>

$$A^t Ay = A^t b. \tag{1.3}$$

Many algorithms for solving geodetic least squares adjustment problems involve the calculation and solution of some (weighted) form of these normal equations (see Ashkenazi [1971] and Bjerhammar [1973]). But because of the size of these problems it is important that particular attention be paid to the numerical stability of the algorithm being used. It is generally agreed in modern numerical analysis theory (see Hanson and Lawson [1974]) that orthogonal decomposition methods applied directly to the matrix A in (1.2) are preferable to the calculation of the normal equations whenever numerical stability is important. Perhaps the most common orthogonal decomposition algorithm in use today is that based upon Householder transformations and developed by Golub [1965].

However, Nerdal [1973] has shown in detail that when computer storage requirements and execution time are taken into account, Golub's method is often inferior even to the use of the normal equations in solving geodetical network problems. It is also known that the modified Gram-Schmidt method has even worse fill-in characteristics and that the modified Givens method shows only slight improvement over the use of Householder transformations for orthogonal decompositions whenever fill-in properties are considered (see Björk [1978] or Duff [1977]). Alternatives to these methods then lead to iterative least squares procedures. Saxena [1972] and Clasen [1977] have applied the conjugate gradient semi-iterative method to least squares geodetical network problems, but each has reported very slow convergence.

The purpose of this paper is to develop two alternative methods that involve neither the ill-conditioned normal equations (1.3) directly or the orthogonal decomposition methods. One of the methods can employ weighting to an advantage and both are simple to formulate and use and are iterative in nature. The obvious advantage with iterative methods is that they usually demand a minimum of storage: only non-zero coefficients of the given matrix need to be stored, plus the coefficients of one or a few vectors such as the solution vector y and, in our case, the residual vector r.

Some comments will be given next about scaling or weighting methods to force convergence or to improve the rate of convergence for iterative methods in solving least squares angle adjustment problems. In geodetical position networks each of the n primary angles formed is measured directly. These measurements are then used in conjunction with the natural relationships among the angles and triangles involved to form the remaining m - n equations of the overdetermined linear system. Since it is these directly measured quantities that are being adjusted, it is quite logical and often convenient to weight rather heavily the equations involving single directly measured angles or, dually, to down-weight the equations involving multiple angles (see Veress [1974]). Such a weighting procedure will be derived and shown to be computationally convenient in forcing convergence of the first iterative method to be employed here.

In Section 2 a 3-block iterative method will be given for the case where the equations in the system (1.2) can be permuted so that the first n rows of the resulting matrix of coefficients form a nonsingular matrix A_1. For the particular application to geodetical position network problems discussed here, A_1 can in fact be chosen to be the identity matrix. A scheme by Chen [1975] is used to apply the block form of A to obtain a block 3-cyclic nonsingular system of linear equations which contains the least squares solution to (1.2). Certain criteria are established for the convergence of the 3-block SOR method for this system and a method for approximating the optimum SOR relaxation parameter is given, providing a rare application of the theory of p-cyclic matrices, with $p \neq 2$, developed by Varga [1959]. In general this 3-block SOR method will not converge unless certain of the equations are weighted. Section 3 is concerned with an alternate, but related, scheme which is based upon the Chebychev semi-iterative method applied to the Gauss-Seidel iterations. No weighting is necessary here. These methods are applied in Section 4 to the iterative solution of the (weighted) least squares adjustment problem associated with geodetical position networks. In Section 5 some comments on the algorithms are given, as well as a discussion of other possible further applications.

2. A Block SOR Least Squares Method. Here a 3-block SOR type iterative procedure used earlier by Chen [1975] and Melendez [1977] will be developed further and applied to the problem at hand.

Suppose an m × n matrix A having full column rank n has been permuted into the form

$$A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$$

where A_1 is n × n and nonsingular (that this can easily be accomplished in the geodetical network problem will be shown in Section 3). Let y be the least squares solution to Ax = b and set r = b - Ay. Partition b and r conformally with A into

$$b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \qquad r = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}$$

so that b_1 and r_1 are n-vectors. Then y is the least squares solution to Ax = b if and only if

$$\begin{aligned} A_1y + r_1 &= b_1 \\ A_2y + r_2 &= b_2 \qquad (2.1) \\ A_2^tr_2 + A_1^tr_1 &= 0. \end{aligned}$$

From these block matrix equations there results the well-known (see Ben-Israel and Greville [1973]) nonsingular system of m + n linear equations in m + n unknowns

$$(2.2)\qquad \begin{bmatrix} A_1 & 0 & I \\ A_2 & I & 0 \\ 0 & A_2^t & A_1^t \end{bmatrix} \begin{bmatrix} y \\ r_2 \\ r_1 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ 0 \end{bmatrix} .$$

This formulation of the least squares problem has also been used by Chen [1975].

Throughout the remainder of the paper the coefficient matrix of (2.2) will be denoted by G, namely,

$$(2.3)\qquad G = \begin{pmatrix} A_1 & 0 & I \\ A_2 & I & 0 \\ 0 & A_2^t & A_1^t \end{pmatrix} .$$

Letting D, L and U denote the block matrices

$$(2.4)\qquad D = \begin{pmatrix} A_1 & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & A_1^t \end{pmatrix},\ L = \begin{pmatrix} 0 & 0 & 0 \\ -A_2 & 0 & 0 \\ 0 & -A_2^t & 0 \end{pmatrix},\ U = \begin{pmatrix} 0 & 0 & -I \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

it follows that the block Jacobi iteration matrix for G is given by

$$(2.5)\qquad J = D^{-1}(L + U) = \begin{pmatrix} 0 & 0 & -A_1^{-1} \\ -A_2 & 0 & 0 \\ 0 & -A_1^{-t}A_2^t & 0 \end{pmatrix} .$$

Since we shall not need the block formulation of the SOR iteration matrix $\mathcal{L}_\omega$ explicitly, it will not be given here. Let

$$x^{(k)} = \begin{pmatrix} y^{(k)} \\ r_2^{(k)} \\ r_1^{(k)} \end{pmatrix}$$

denote the k^th approximation vector to the vectors y, r_2 and r_1. Then with D given by (2.4) and J by (2.5), the Jacobi method

$$x^{(k+1)} = Jx^{(k)} + D^{-1}b, \quad k = 0, 1, 2, \ldots$$

has the block matrix form:

$$y^{(k+1)} = A_1^{-1}(b_1 - r_1^{(k)})$$

$$r_2^{(k+1)} = -A_2 y^{(k)} + b_2$$

$$r_1^{(k+1)} = -A_1^{-t} A_2^t r_2^{(k)}$$

for k = 0, 1, 2, ...

In practical computation one is often not concerned so much with the block Jacobi method but rather with the block SOR iterative method. The block SOR iterative method for the system (2.2) is obtained in a fashion similar to the derivation of the block Jacobi method. In particular, the SOR method for $\omega > 0$ and $\mathcal{L}_\omega = (D - \omega L)^{-1}[(1 - \omega)D + \omega U]$, given by

$$x^{(k+1)} = \mathcal{L}_\omega x^{(k+1)} + (D - \omega L)^{-1} b,$$

has the block matrix form

$$y^{(k+1)} = y^{(k)} + \omega[A_1^{-1}(b_1 - r_1^{(k)}) - y^{(k)}] \tag{2.6}$$

$$r_2^{(k+1)} = r_2^{(k)} + \omega[b_2 - A_2\, y^{(k+1)} - r_2^{(k)}] \tag{2.7}$$

$$r_1^{(k+1)} = r_1^{(k)} + \omega[-A_1^{-t}A_2^t r_2^{(k+1)} - r_1^{(k)}] \tag{2.8}$$

for k = 0, 1, 2, ...

As with the block Jacobi method, the matrices A_1^{-1} and $A_1^{-t}A_2^t$ need not be computed and formed explicitly in this process.

The underlying problem in all linear stationary iterative processes is that of establishing convergence. There are basically two classes of coefficient matrices for which convergence criteria are readily available: symmetric positive definite matrices and M-matrices (see Berman and Plemmons [1979], Chapter 7, or Varga [1962]). Since the coefficient matrix G given by (2.3) can never be either of these two types for m > n, alternative convergence criteria need to be derived and this is accomplished next, with application to the geodetical network problem in mind.

The theory of p-cyclic matrices developed by Varga [1959] can be used to study the convergence of the block SOR method (2.6) - (2.8) for the general problem. Moreover, an expression for the optimum relaxation parameter ω_b can be given. In order that these results hold, it is sufficient (see Varga [1959]) that

i) the coefficient matrix be block consistently ordered and p-cyclic,

ii) the eigenvalues of J^p, J the Jacobi iteration matrix, be real and all nonnegative or all nonpositive, and

iii) $\rho(J) < 1$, where $\rho(J)$ is the spectral radius.

Chen [1975] has shown that G given in (2.3) is a consistently ordered 3-cyclic matrix and that all the eigenvalues of J^3 are nonpositive. Thus this situation provides a rare application of Varga's theory for $p \neq 2$.

With this in mind, the study of the convergence of the 3-block SOR method (2.6) - (2.8) is basically reduced to the study of conditions under which $\rho(J) < 1$. Notice first that J given by (2.5) satisfies

$$J^3 = \begin{bmatrix} -A_1^{-1}A_1^{-t}A_2^tA_2 & 0 & 0 \\ 0 & -A_2A_1^{-1}A_1^{-t}A_2^t & 0 \\ 0 & 0 & -A_1^{-t}A_2^tA_2A_1^{-1} \end{bmatrix}.$$

It follows that

$$\rho(J)^3 = \rho[(A_2A_1^{-1})^t(A_2A_1^{-1})] = ||A_2A_1^{-1}||_2^2. \tag{2.9}$$

Thus

$$\rho(J) < 1 \iff ||A_2A_1^{-1}||_2 < 1. \tag{2.10}$$

This leads to a useful upper bound on $\rho(J)$ in terms of the column and row sum norms $||A_2||_1$ and $||A_2||_\infty$, respectively.

(2.11) <u>Lemma</u>. Let $A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$, where A_1 is square and strictly row and strictly column diagonally dominant. Let

$$(2.12) \qquad r = \min_i \{|a_{ii}| - \sum_{j \neq i} |a_{ij}|\}, \; c = \min_j \{|a_{jj}| - \sum_{i \neq j} |a_{ij}|\}.$$

Then

$$(2.13) \qquad \rho(J) \leq \left[\frac{||A_2||_1 \, ||A_2||_\infty}{rc} \right]^{\frac{1}{3}} .$$

<u>Proof</u>. Varga [1976] has shown that under these conditions on A_1,

$$||A_1^{-1}||_2^2 \leq \frac{1}{rc}$$

where r and c are given by (2.12). Then by (2.9) it follows that

$$\rho(J)^3 \leq ||A_2||_2^2 ||A_1^{-1}||_2^2 \leq \frac{||A_2||_2^2}{rc} \leq \frac{||A_2||_1 ||A_2||_\infty}{rc}$$

and thus (2.13) holds. ■

This lemma then gives an obvious sufficient condition for $\rho(J) < 1$. This and other results on the block SOR method (2.6) - (2.8) are given next.

(2.14) <u>Theorem</u>. Let $A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$ where A_1 is strictly row and strictly column diagonally dominant and let r and c be given by (2.12). If

$$(2.15) \qquad ||A_2||_1 ||A_2||_\infty < rc$$

then the block SOR method (2.6) - (2.8) converges to the least squares solution y to Ax = b and the residual vector r = b - Ay, for each relaxation parameter ω satisfying

$$(2.16) \qquad 0 < \omega < \frac{3}{2}.$$

Moreover, with J given by (2.5) the optimum relaxation parameter ω_b is given by

$$(2.17) \qquad \omega_b = \frac{3}{2\rho(J)}\left(\sqrt[3]{1 + \sqrt{1 + \frac{1}{\rho(J)^3}}} + \sqrt[3]{1 - \sqrt{1 + \frac{1}{\rho(J)^3}}}\right)$$

Proof. As indicated earlier, G given by (2.3) is consistently ordered and 3-cyclic and the eigenvalues of the Jacobi iteration matrix are all real and nonpositive. From (2.13) and (2.15) it follows that $\rho(J) < 1$ and thus the theory of Varga [1959] can be applied here. It follows that $\rho(\mathcal{L}_\omega) < 1$ whenever $0 < \omega < \frac{p}{p-1}$ and so (2.16) holds with p = 3. Moreover, it is known (see Varga [1962], Section 4.3) that ω_b is the smallest positive root of the equation

$$-[\rho(J)\omega]^p = p^p(p-1)^{1-p}(\omega-1)$$

which in this case reduces to

$$\omega^3 + \frac{27}{8\rho(J)^3}\omega - \frac{27}{4\rho(J)^3} = 0.$$

It can be shown then that the smallest positive root of this equation is given by (2.17). ■

An important observation from (2.17) is that ω_b satisfies in this case

$$.9 < \omega_b \leq 1.0$$

and for $\rho(J) = 0$, $\omega_b = 1$ and $\omega_b \to .9$ as $\rho(J) \to 1$. This is in contrast to the 2-cyclic case where $\omega_b \geq 1$. Moreover, from Lemma 2.11, the number

$$(2.18) \qquad \sqrt[3]{\frac{||A_2||_1 \, ||A_2||_\infty}{rc}}$$

is a simple upper bound for $\rho(J)$. Also note that the conclusions of Theorem 2.14 still remain valid under the more general hypothesis that $\rho(J) < 1$.

The 3-block SOR least squares algorithm can be summarized in the following form:

(2.19) Algorithm. This algorithm can be used to approximate the least squares solution y to $Ax = b$ and the residual vector $r = b - Ay$, where A has the block form $A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$, with A_1 nonsingular, and $\rho(J) < 1$, where J is given by (2.5).

1. Estimate $\rho(J)$, say by the expression given by (2.18) if A_1 is strictly row and strictly column diagonally dominant.

2. Estimate the optimum SOR parameter ω_b by, say, (2.17).

3. Choose initial approximations $y^{(0)}$ and $r^{(0)} = \begin{bmatrix} r_1^{(0)} \\ r_2^{(0)} \end{bmatrix}$ (arbitrarily) to the least squares solution and the residual vector, respectively.

4. For k = 0, 1, 2, ...

   a. Compute $y^{(k+1)}$ by (2.6). Here an inner iteration procedure or a sparse matrix decomposition should be used to compute the vector $A_1^{-1}(b_1 - r_1^{(k)})$.

   b. Compute $r_2^{(k+1)}$ by (2.7).

   c. Compute $r_1^{(k+1)}$ by (2.8). Here the matrix $A_1^{-t}A_2^t$ need not be formed explicitly and an inner iteration procedure or a sparse matrix decomposition should be used to compute the vector $-A_1^{-t}A_2^t r_2^{(k+1)}$.

An adaptation of this algorithm to solve the least squares adjustment problem for geodetical networks will be given in Section 4.

In most practical situations, including the angle adjustments problem, one would not normally expect the spectral radius of the Jacobi iteration matrix to be less than one. In this case Algorithm 2.19 cannot be applied directly to the problem in question. In the next section this convergence difficulty is overcome by use of the Chebyshev semi-iterative method.

3. Chebyshev Acceleration. It will be demonstrated in this section how the Chebyshev semi-iterative method can be combined with the 3-block iterative method of Section 2, to produce a sequence of vectors which always converges to the solution, $(y,r_2,r_1)^t$, to the 3-block form of the least squares problem given by (2.2). For that purpose it will be convenient to briefly review the Chebyshev method for accelerating the convergence of a sequence.

Following Concus, Golub and O'Leary [1976] and Young [1971], suppose we have a basic iterative scheme

$$(3.1) \qquad x^{(k+1)} = Bx^{(k)} + c \quad , \; k = 0, 1, \ldots$$

where the eigenvalues λ of the iteration matrix B are all real and lie in the range

$$m \leq \lambda \leq M < 1 \quad , \quad m \neq M \; .$$

Then the result of applying the 3-term Chebyshev semi-iterative method to (3.1) is

$$(3.2) \qquad u^{(k+1)} = \beta_{k+1}\left[\frac{2}{2-(M+m)}\left(Bu^{(k)} + c - u^{(k)}\right) + u^{(k)}\right] + (1-\beta_{k+1})\,u^{(k-1)}$$

where for $\alpha = \dfrac{M-m}{2-(M+m)}$,

$$\beta_1 = 1 \; , \; \beta_2 = (1 - \tfrac{1}{2}\alpha^2)^{-1} \; , \; \beta_{k+1} = (1 - \tfrac{1}{4}\alpha^2\beta_k)^{-1} \; , \; k \geq 1 \; .$$

It follows (e.g. Young [1971], Chapter 11) that under these conditions on B, the sequence $\{u^{(k)}\}$ generated by (3.2) always converges to $x = (I - B)^{-1}c$, even though the sequence generated by (3.1) may not converge.

The Chebyshev method will now be applied to the 3-block SOR scheme described in Section 2. For that purpose we choose to accelerate the Gauss-Seidel scheme; that is, the SOR scheme given by (2.6) - (2.8) with $\omega = 1$. Now with D, L and U given by (2.4), it follows that the Gauss-Seidel iteration matrix $\mathcal{L}_1$ for the 3-block system (2.2) is

$$\text{(3.3)} \quad \mathcal{L}_1 = (D - L)^{-1}U = \begin{bmatrix} 0 & 0 & -A_1^{-1} \\ 0 & 0 & A_2A_1^{-1} \\ 0 & 0 & -(A_2A_1^{-1})^t(A_2A_1^{-1}) \end{bmatrix}$$

Moreover the basic 3-block Gauss-Seidel scheme to be accelerated reduces, from (2.6)-(2.8) with $\omega = 1$, to the simple form

$$y^{(k+1)} = A_1^{-1}(b_1 - r_1^{(k)}) \ ,$$

$$r_2^{(k+1)} = b_2 - A_2y^{(k+1)} \quad ,$$

$$r_1^{(k+1)} = -(A_2A_1^{-1})^t r_2{}^{(k+1)} .$$

This avoids the direct application of the matrix $\mathcal{L}_1$ in the iterative scheme.

Now from (3.3), the eigenvalues of $\mathcal{L}_1$ are those of $-(A_2A_1^{-1})^t(A_2A_1^{-1})$ which are real and nonpositive and, in fact, satisfy

$$-\rho \leq \lambda \leq 0$$

where

$$\text{(3.4)} \quad \rho = \rho[(A_2A_1^{-1})^t(A_2A_1^{-1})] = ||A_2A_1^{-1}||_2^2 \ .$$

Thus the Chebyshev semi-iterative scheme (3.2) can be applied to the basic iterative scheme with $M = 0$ and $m = -\rho$. A slightly modified form of the method is summarized as follows:

(3.5) Algorithm. This algorithm can be used to approximate the least squares solution y to $Ax = b$ and the residual vector $r = b - Ay$, where A has the block form $A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$, with A_1 nonsingular.

1. Estimate the parameter ρ given by (3.4).

2. Choose initial approximations $y^{(k)}$ and $r^{(k)} = \begin{bmatrix} r_1^{(k)} \\ r_2^{(k)} \end{bmatrix}$ to the least squares solution y and the residual vector r , partitioned as A , for $k = -1 , 0$.

3. Perform the following steps for $k = 0, 1, 2, \ldots$:

(a) Compute $y^{(k+1)}$ from

$$y^{(k+1)} = \beta_{k+1}\left[\frac{2}{2+\rho}(A_1^{-1}(b_1 - r_1^{(k)}) - y^{(k)}) + y^{(k)}\right] + (1-\beta_{k+1})y^{(k-1)} , \tag{3.6}$$

(b) Compute $r_2^{(k+1)}$ from

$$r_2^{(k+1)} = \beta_{k+1}\left[\frac{2}{2+\rho}(b_2 - A_2y^{(k+1)} - r_2^{(k)}) + r_2^{(k)}\right] + (1-\beta_{k+1})r_2^{k-1} , \tag{3.7}$$

(c) Compute $r_1^{(k+1)}$ from

$$r_1^{(k+1)} = \beta_{k+1}\left[\frac{2}{2+\rho}(-(A_2A_1^{-1})^t r_2^{(k+1)} - r_1^{(k)}) + r_1^{(k)}\right] + (1-\beta_{k+1})r_1^{(k-1)} \tag{3.8}$$

where

$$\beta_1 = 1 , \ \beta_2 = \left[1 - \tfrac{1}{2}\left(\frac{\rho}{2+\rho}\right)^2\right]^{-1} , \ \beta_{k+1} = \left[1 - \tfrac{1}{4}\left(\frac{\rho}{2+\rho}\right)^2\beta_k\right]^{-1} , \ k \geq 2 .$$

This algorithm always converges, then, tc the vectors y and r .

Adaptations of Algorithms (2.19) and (3.5) to solve the least squares angle adjustment problem for geodetical networks are given in the next section.

4. Geodetical Angle Adjustments. The purpose of this section is to apply the 3-block methods developed in Sections 2 and 3 to least squares angle adjustments in geodetical network problems. Consider first the geodetical position network given by Figure 1. The particular form of the coefficient matrix in the overdetermined system (1.1) associated with this problem does not readily fit our needs. However, suppose the equations are permuted so that the constant vector b has the form, say,

$$b^t = [m_1, m_2, m_3, m_4, m_5, m_6, m_7, m_8, m_9, 180°, 180°, 180°, \alpha, \beta, \gamma, \delta, \varepsilon],$$

so that the measurements of the primary angles come first and are ordered m_1 through m_9 (this can also be accomplished by an appropriate renumbering of the angles in the network). Then the coefficient matrix A of the resulting overdetermined system of linear equations has the form

$$(3.1) \qquad A = \begin{bmatrix}
x & & & & & & & & \\
& x & & & & & & & \\
& & x & & & & & & \\
& & & x & & & & & \\
& & & & x & & & & \\
& & & & & x & & & \\
& & & & & & x & & \\
& & & & & & & x & \\
& & & & & & & & x \\
x & x & & & x & x & & & \\
& x & x & x & x & & x & & \\
& & & x & & & x & x & x \\
& x & x & & & & & & \\
& & & & & & x & x & \\
& & & x & x & & & & \\
& & & & x & x & & & \\
& & & x & & x & & &
\end{bmatrix}$$

where "x" denotes a nonzero entry. Here of course A has the block partitioned form $A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$ where $A_1 = I$, the identity matrix of order 9. In general such a permutation is always possible so that A_1 is the identity matrix of order n, n being the number of primary angles in the network. It will be assumed that such a permutation has been made in the remainder of the paper. The 3-block SOR method of Section 2 will be discussed first.

Now A_1 is strictly row and strictly column diagonally dominant and r = c = 1 in (2.12). However, A_2 clearly does not satisfy (2.5) and $||A_2||_2 > 1$ so that Theorem 2.14 does not apply here; and, in fact, the 3-block SOR method does not converge for all $0 < \omega < \frac{3}{2}$ by (2.10). This will generally be the case for the geodetical network angle adjustment problem before the equations involving multiple angles are down-weighted. It turns out that a judicious choice of a scaling factor for the equations corresponding to A_2 leads to a situation where all the hypotheses of Theorem 2.14 are satisfied.

As indicated in Section 1, since direct measurements are being adjusted, equations involving multiple angles may be down-weighted before solving the least squares adjustment problem. For our purpose the scaling factor

$$s = \frac{1}{||A_2||_1 ||A_2||_\infty} \tag{4.2}$$

will be convenient. In this case the replacements

$$\tilde{A}_2 = sA_2, \quad \tilde{b}_2 = sb_2 \tag{4.3}$$

are made prior to solving the least squares problem. Thus since $r = c = 1$ in (2.12),

$$||\tilde{A}_2||_1 ||\tilde{A}_2||_\infty = s < 1 = rc$$

so that hypotheses of Theorem 2.14 are now satisfied.

Of course in practice the scaling factor s should always be chosen so that $s < \sqrt{EPS}$, where EPS is the precision of the particular machine being used (e.g., Bjӧrk [1978]). It turns out that the scaling factor s given by (4.2) can easily be bounded.

Consider any geodetical network angle adjustment problem where the matrix of coefficients has the partitioned form,

$A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$, $A_1 = I$. Then $||A_2||_\infty$ is the maximum number of angles involved in any equation involving multiple angles and this maximum occurs then in the 180° angle sum equations. Clearly then $||A_2||_\infty = 3$ when the network has $t = 3$ stations and $||A_2||_\infty \geq 4$ when $t > 3$. Also, $||A_2||_\infty$ is maximal when the network has an embedded configuration of the form

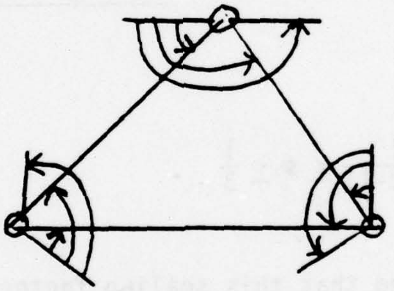


Figure 3

In this case $||A_2||_\infty = 6$ and we have

$$3 \leq ||A_2||_\infty \leq 6$$

for any network. In order to bound $||A_2||_1$, note first that $||A_2||_1 = 1$ when $t = 3$ stations. Moreover, a single primary angle in a network can be involved in at most two 180° angle sum equations. However, for $t > 3$, an angle may be involved in up to t - 3 equations involving the difference of primary angles, so that $||A_2||_1 \leq 2 + t - 3 = t - 1$. Consequently we have the bounds

$$1 \leq ||A_2||_1 \leq t - 1,$$

for any network. Thus

$$3 \leq ||A_2||_\infty ||A_2||_1 \leq 6(t - 1)$$

for any network with t stations and so the scaling factor s given by (4.2) satisfies the bounds

(4.4) $$\frac{1}{6(t-1)} \leq s \leq \frac{1}{3} .$$

It should be emphasized that this scaling factor will not be unrealistically small for most networks of practical interest and that the upper bound t - 1 for $||A_2||_1$ will be reached only in very special networks. By Theorem 2.14, the 3-block SOR Algorithm 2.19 converges. Moreover, $\sqrt[3]{5}$ gives a convenient upper bound for $\rho(J)$ by 2.18.

The 3-block SOR least squares method for geodetical angle adjustments is summarized as follows:

(4.5) Algorithm. This algorithm can be used to compute weighted least squares adjustments to angle measurements in geodetical network problems.

1. Number the station angles so that $A_1 = I$ and determine the resulting observation vector $b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$, b_1 an n-vector.

2. Form the equations involving the multiple angles and construct the coefficient matrix A_2.

3. Compute the scaling factor $s = \frac{1}{||A_2||_1||A_2||_\infty}$ and make the replacements (4.3), forming $\tilde{A}_2$ and $\tilde{b}_2$.

4. Apply Algorithm 2.19 to compute the weighted least squares solution vector y and the residual vector r for the scaled system of linear equations. Then y gives the adjusted angle measurements associated with the geodetical network in question.

Some comments on the implementation of step 4 of Algorithm 4.5 are in order. Notice that, since $A_1 = I$, the 3-block SOR algorithm given in Section 2 is simplified considerably. In particular, equation (2.6) simplifies to

$$y^{(k+1)} = (1 - \omega)y^{(k)} + \omega(\tilde{b}_1 - r_1^{(k)})$$

and equation (2.8) simplifies to

$$r_1^{(k+1)} = (1 - \omega)r_1^{(k)} - \omega\tilde{A}_2^t r_2^{(k+1)},$$

while equation (2.7) remains the same. Also note that $\rho(J)$ can be estimated in step 1 of Algorithm 2.19 by the simple expression $\sqrt[3]{s}$ where s is given by (4.2).

Some numerical experiments involving Algorithm 4.5 will be discussed next.

Algorithm 4.5 has been successfully used to solve some least squares angle adjustment problems associated with geodetical networks. In particular, several tests were made of such problems using a DEC-10 computer with a wordlength corresponding to approximately 8 decimal figures (EPS $\approx .37 \times 10^{-8}$) at the University of Tennessee Computing Center. A FORTRAN IV implementation of Algorithm 4.5 which takes advantage of the special structure of the matrix of coefficients was used. For example the locations of the nonzero elements (1 or -1) of the matrix were stored in lists by rows. This is particularly advantageous in step 4 in which the 3-block SOR algorithm (in its simplified form) is involved. A stopping criteria based in part upon the smallness of $||A^t r^{(k)}||_2^2$ was used because the k^{th} residual vector $r^{(k)}$ is readily available in the iterative sequence and because this is a commonly used stopping criteria for other iterative least squares adjustment algorithms (see Saxena [1972] and Clasen [1977]). The iterations were terminated if either

$$||A^t r^{(k)}||_2^2 < 10^{-6} \text{ or } \frac{||y^{(k)} - y^{(k-1)}||_2^2}{||y^k||_2^2} < 10^{-6}.$$

As a first example the geodetic position network with $t = 5$ stations, given in Figure 1, was considered. Here an appropriate choice of angle observations was made and the scaling factor $s = \frac{1}{||A_2||_1 \, ||A_2||_\infty} = \frac{1}{20}$ was used to obtain the approximation $\rho(J) \approx \sqrt[3]{\frac{1}{20}}$.

The optimum SOR parameter ω_b given by (2.17) is then approximated by $\omega_b \approx .9796$. Using these estimates, together with the initial approximations $y^{(0)} = r^{(0)} = 0$ in Algorithm 2.19, the 3-block SOR method specified by step 4 was used to approximate the weighted least squares solution to the angle adjustment problem. The stopping criteria $||A^t r_2^{(k)}||_2^2 < 10^{-6}$ was satisfied after $k = 6$ iterations. It should be mentioned that the stopping criteria was reached after only $k = 5$ iterations with the choice of $\omega = 1$. This was expected, however, since the analysis (see Varga [1959]) of the 3-cyclic case shows that it is better to overestimate ω_b. For this simple problem it appears that the true value of ω_b was closer to 1.

Next, a geodetical network with $t = 100$ stations was considered. From Table 1 we see that there are 294 primary angles to be adjusted. Thus the resulting overdetermined system of linear equations $Ax = b$ has $n = 294$ unknowns. For this example there were $m = 538$ equations. As before, an appropriate choice of primary angle observations was made. For this particular problem $||A_2||_1 = 7$ and $||A_2||_\infty = 6$, so that the scaling factor is given by $s = \frac{1}{42}$. Algorithm 4.5 was then applied to the problem and the approximation $\hat{\rho}(J) \approx \sqrt[3]{\frac{1}{42}}$ was used in Algorithm 2.19. Here the convergence criteria was reached after only $k = 206$ iterations, and $k = 217$ iterations were needed for the Gauss-Siedel method, i.e., $\omega = 1$. This compares quite favorably with results reported in Clasen [1977] and Saxena [1972] where approximately $2n$ iterations were required to produce similar accuracy using the conjugate gradient method applied to the normal equations.

If weighting the equations in order to apply Algorithm 4.5 is not acceptable to the user, then an alternative procedure is to apnly the Chebyshev semi-iterative method, Algorithm 3.5, to the 3-block Gauss-Seidel iterations $(\omega = 1)$. No weighting is necessary for convergence here and an estimate for the parameter ρ involved is readily available.

Since we may number the station angles so that $A_1 = I$, it follows from (3.4) and the analysis preceeding Algorithm 4.5 that

$$\rho = \rho(A_2^t A_2) = ||A_2||_2^2 \leq ||A_2||_\infty \; ||A_2||_1 \leq 6(t-1) ,$$

for any network with t stations. Since it is better to underestimate the smallest eigenvalue, $-\rho$, of the iteration matrix (see Young [1971], Chapter 11), the simple-to-compute value $||A_2||_1 \; ||A_2||_\infty$ may in certain cases be an acceptable estimate for ρ.

The Chebyshev semi-iterative method applied to the 3-block Gauss-Seidel iterations for angle adjustments is summarized as follows:

(4.6) Algorithm. This algorithm can be used to compute (unweighted) least squares adjustments to angle measurements in geodetical network problems.

1. Number the station angles so that $A_1 = I$ and determine the resulting observation vector $b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$, b_1 an n-vector.

2. Form the equations involving the multiple angles and construct the coefficient matrix A_2.

3. Approximate $\rho = \rho[(A_2A_1^{-1})^t(A_2A_1^{-1})] = ||A_2||_2^2$ by, say, $||A_2||_1 \; ||A_2||_\infty$.

4. Apply Steps 2 and 3 of Algorithm 3.5 to compute the least squares solution vector y and the residual vector r for the system of linear equations. Then y gives the adjusted angle measurements associated with the geodetical network in question.

As was the case of the 3-block SOR algorithm, the fact that $A_1 = I$ in Algorithm 4.6 enables a simplification of the iterative formulas. Here equation (3.6) simplifies to

$$y^{(k+1)} = \beta_{k+1}\left[\frac{2}{2+\rho}(b_1 - r_1^{(k)}) + y^{(k)}\right] + (1-\beta_{k+1})y^{(k-1)}$$

and equation (3.8) simplifies to

$$r_1^{(k+1)} = \beta_{k+1}\left[\frac{2}{2+\rho}(-A_2^t r_2^{(k+1)} - r_1^{(k)}) + r_1^{(k)}\right] + (1-\beta_{k+1})r_1^{(k-1)} ,$$

while equation 3.7 remains the same.

Since Algorithm 4.6 computes the general unweighted least squares solution while Algorithm 4.5 computes a weighted least squares solution, no direct numerical comparisons of the two methods are possible. It should be pointed out that Algorithm 4.6 proved to be slightly easier to implement while convergence was slower than that of the 3-block SOR method in each case, as one might expect since no weighting is involved in the Chebyschev semi-iterative method. However, fewer than n iterations were required to reach the stopping criteria given earlier; and, once again, this compares quite favorably with the conjugate gradient method applied to the normal equations.

5. Observations. When the present state of the art for computational least squares methods for large sparse systems of linear equations is considered, the 3-block SOR method and the Chebyshev semi-iterative method applied to the 3-block Gauss-Seidel iterations appear to be quite promising for certain adjustment problems. For the least squares angle adjustment problem in Geodesy, these methods are especially attractive due to the particular structure of the matrix of coefficients. When computer storage considerations and execution time are important, then these methods are to be preferred over those involving orthogonal decompositions or the direct formulation of the normal equations; for here the matrix of coefficients A has at most 6 nonzero entries in each row (+1 or -1) no matter how many stations are involved in the network, and A is not altered in these iterative methods.

If the user is willing to accept the weighting necessary to guarantee convergence of the 3-block SOR method, then Algorithm 4.5 profides a slightly more efficient method for the solution to the least squares angle adjustment problem. If the weighting scheme is not acceptable, then the Chebyshev semi-iterative applied to the Gauss-Seidel iterations given in Algorithm 4.6, is recommended. Computational experience also indicates that an estimate for $\rho = ||A_2||_2^2$ more accurate than $||A_2||_1 \; ||A_2||_\infty$ is usually not necessary. An efficient method for computing this largest singular value of A_2 by use of the block Lancoz algorithm is given by Golub, Luk and Overton [1977].

Iterative schemes similar to those given in this paper can be devised so that the residual vectors are not directly involved. In particular, suppose $A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$ with A_1 nonsingular. Then

$$y^{(k+1)} = -(A_1^tA_1)^{-1}(A_2^tA_2)\; y^{(k)} + A^tb$$

defines an iterative scheme which, if convergent, will converge to the least squares solution y to (1.1). However, this scheme was not seriously considered here since an implicit formulation of certain normal equations is involved and, also, the residual vector $r^{(k)} = b - Ay^{(k)}$ is useful in stopping criteria for the iteration. Moreover, least squares iterative schemes, such as iterative refinement, generally give better results whenever the residual vectors are used (See Björk [1976, 1978]).

Another promising application of the 3-block SOR algorithm given in Section 2 or the Chebyschev semi-iterative method given in Section 3, lies in the area of least squares coordinate adjustments for large geodetic triangulation systems. Such systems are important, for example, in the triangulation of planitary surface coordinates (see Davies and Author [1973] and Saxena [1972]) and for the analysis of large amounts of satelite photographic data. Here the coefficient matrix A for the linear system, in general, is $m \times n$ $(m >> n)$ where m is the number of observations and n is the number of unknowns, which is equal to three times the number of stations since for each station there are three unknowns: the orientation correction and two coordinate corrections. Each row of A contains a maximum of five nonzero elements $(||A||_\infty = 5)$ where one term is the coefficient of the orientation correction and the other four terms are the coefficients of station coordinate corrections. Some further investigation of this application is planned.

REFERENCES

V. Ashkenazi [1971], "Geodetic normal equations," in Large Sets of Linear Equations, ed. J. K. Reid, 57-74, Academic Press, N. Y.

A. Ben-Israel and T. N. E. Greville [1974], Generalized Inverses, Theory and Applications, John Wiley and Sons, N. Y.

A. Berman and R. J. Plemmons [1979], Nonnegative Matrices in the Mathematical Sciences, Academic Press, N. Y.

A. Bjerhammar [1973], Theory of Errors and Generalized Matrix Inverses, Elsevier, Amsterdam.

Å. Björk [1976], "Methods for sparse linear least squares problems," in Sparse Matrix Computations, ed. J. R. Bunch and D. J. Rose, 177-194, Academic Press, N. Y.

Å. Björk [1978], "Numerical algorithms for linear least squares problems," Math. and Computation Rept. 2/78, Dept. of Math., Univ. of Trondheim, Trondheim, Norway.

Å. Björk and T. Elfving [1978], "Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations," Report Lith-Mat.-R-1978-5, Linköping Univ., Linköping, Sweden.

Y. T. Chen [1975], "Iterative methods for linear least squares problems," Ph.D. dissertation, Dept. Comp. Sci., Univ. Waterloo, Waterloo, Ontario, Canada.

R. J. Clasen [1977], "A note on the use of the conjugate gradient method in the solution of a large system of sparse equations," Computer J. 20, 185-186.

P. Concus, G.H. Golub and D.P. O'Leary [1976], "A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations", in Sparse Matrix Computations, ed. J.R. Bunch and D.J. Rose, 309-332, Academic Press, NY.

M. E. Davies and D. W. G. Arthur [1973], "Martian surface coordinates," J. Geophys. Res., 78, 4355-4394.

I. S. Duff [1977], "A survey of sparse matrix research," Proc. of the IEEE, 65, 500-535.

T. Elfving [1977], "Group-iterative methods for consistent and inconsistent linear equations," Rept.Lith.-Mat.-R-1977-11, Linköping Univ., Linköping, Sweden.

T. Elfving [1978], "On the conjugate gradient method for solving linear least squares problems," Rept. Lith.-Mat.-R-1978-3, Linköping Univ., Linköping, Sweden.

G. H. Golub [1965], "Numerical methods for solving linear least squares problems," Numer. Math., 7, 206-216.

G.H. Golub, F.T. Luk and M.L. Overton [1977], "A block Lancoz method to compute the singular values and corresponding singular vectors of a matrix," Rept. Stan-CS-77-635, Stanford, CA.

R.J. Hanson and C.L. Lawson [1974], Solving Least Squares Problems, Prentice-Hall, Inc., Englewood Cliffs, N.J.

G. Melendez [1977], "Block iterative methods for large sparse least squares problems," M.S. thesis, Math. Dept., Univ. of Tn., Knoxville, Tn.

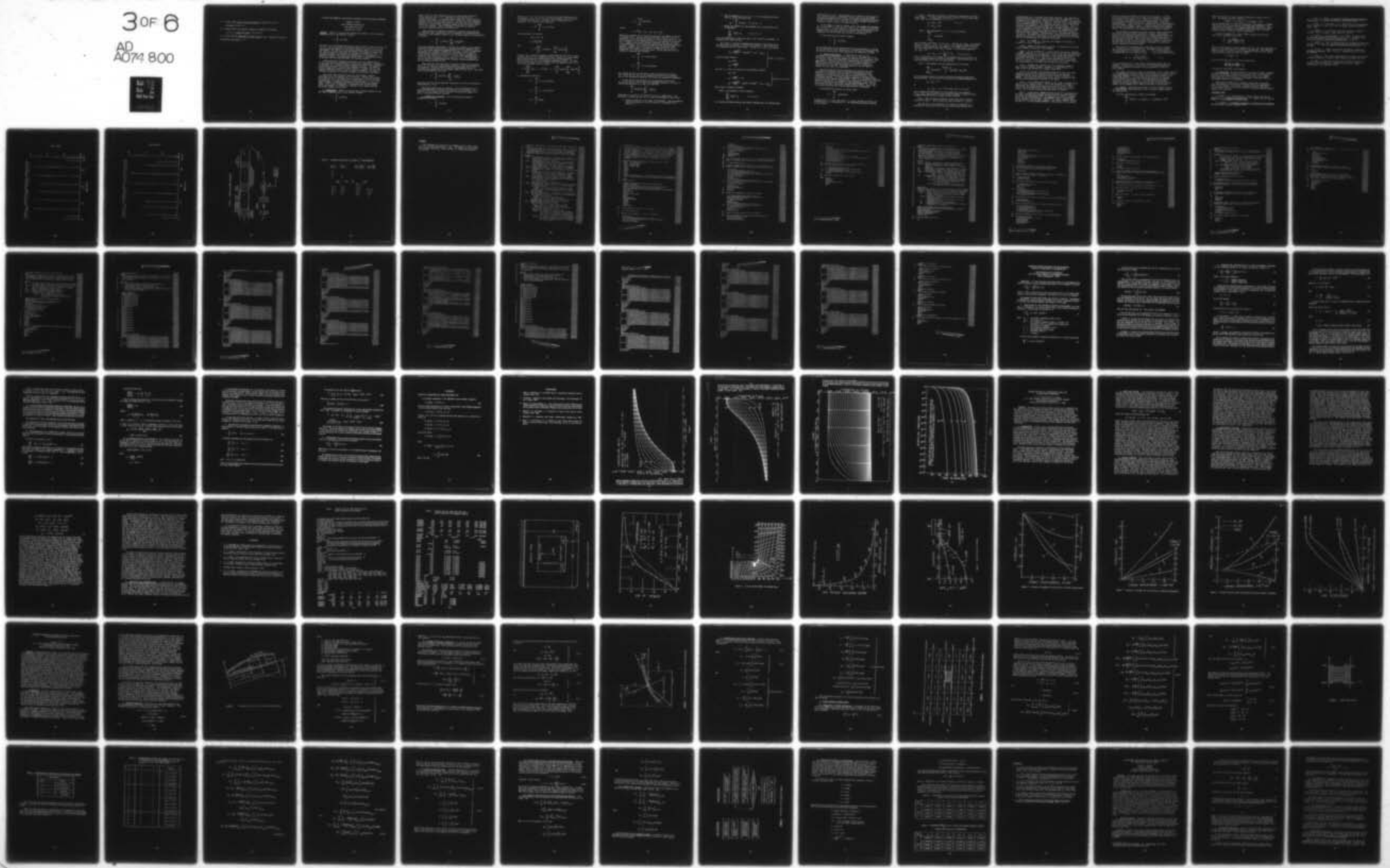
O. Nerdal [1973], "Sparse matrix techniques to solve geodetical network problems," Rept. Uminf-39.73, Univ. Umeå, Umeå, Sweden.

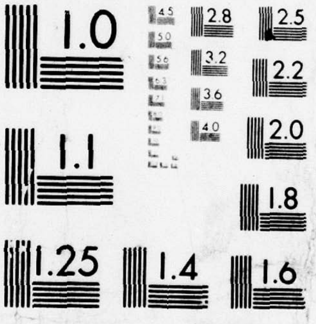
N. K. Saxena [1972], "Improvement of a geodetical triangulation through control points established by means of satellites or precision traversing," Rept. 177, Dept. of Geod. Sci., Ohio State U., Columbus, Ohio.

R. S. Varga [1959], "p-cyclic matrices: a generalization of the Young-Frankel successive overrelaxation scheme," Pac. J. Math., 9, 617-628.

AD-A074 800 ARMY RESEARCH OFFICE RESEARCH TRIANGLE PARK NC F/G 5/9
PROCEEDINGS OF THE 1979 ARMY NUMERICAL ANALYSIS AND COMPUTERS C--ETC(U)
SEP 79
UNCLASSIFIED ARO-79-3 NL

3 OF 6
AD
A074 800




1.0
4.5
5.0
5.6
6.3
7.1
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

R. S. Varga [1962], Matrix Iterative Analysis, Prentice-Hall, Inc., Englewood Cliffs, N. J.

R. S. Varga [1976], "On diagonal dominance arguments for bounding $||A^{-1}||_{\infty}$," Linear Alg. Appl., 14, 211-217.

S. A. Veress [1974], Adjustment by Least Squares, Amer. Congress on Surveying and Mapping, Washington, D. C.

A ROUTINE FOR NUMERICAL EVALUATION OF INTEGRALS WITH OSCILLATORY INTEGRANDS

Theodore H. Hopp
Harry Diamond Laboratories
2800 Powder Mill Road
Adelphi, Md. 20783

ABSTRACT. IFEIX is a routine that provides an estimate of the real and/or imaginary parts of the definite integral

$$I = \int_a^b f(x)e^{ix}dx,$$

where f(x) is a real-valued function of a real variable, a and b are real, and $i = \sqrt{-1}$. The user specifies the desired absolute or relative error of the approximation and whether the real and/or imaginary parts of the integral are desired. The user also specifies whether the program can assume that the lower boundary of integration is an even or odd multiple of $\pi/2$. For applications where only the real or imaginary part of the integral is desired (or exists), this indication allows IFEIX to successfully provide an estimate when the algorithm might not otherwise succeed.

IFEIX operates by dividing the interval [a,b] into subintervals of no more than 640π and integrating each subinterval separately. Integration is performed via a 10-point Gaussian integration scheme using quadrature coefficients developed with suitable weighting functions. An interval halving scheme with an interval queue (similar to schemes used in other popular integration codes) is used to obtain convergence. An Aitken δ^2 transform is applied to successive estimates to accelerate convergence.

Since IFEIX uses Gaussian integration coefficients, endpoint singularities of the integrand can be tolerated. IFEIX has been tested for a wide variety of integrands and has been shown to be both reliable and efficient. For large intervals of integration ($|b-a| >> 2\pi$), IFEIX exhibits orders of magnitude improvement in performance over existing codes. IFEIX is not recommended for integrands with discontinuities within the interval of integration, although it will usually provide the correct answer in such cases.

1. INTRODUCTION.

IFEIX is a routine that provides estimates of the real and/or imaginary parts of the definite integral

$$I = \int_a^b f(x)e^{ix}dx,$$

where a and b are real and finite, f(x) is a real-valued function of a real variable, and $i = \sqrt{-1}$. Integrals of this form arise in various applications: for instance, Fourier analysis, various mechanics problems, and problems of physical optics. Quite often these integrals cannot be solved in closed form, while, at the same time, their numerical evaluation is prohibitively expensive because of the oscillatory factor e^{ix} in the integrand. IFEIX is designed to overcome the numerical efficiency problems created by the factor e^{ix}.

Most software for numerical integration is based on approximating the integrand under consideration with a polynomial and integrating that polynomial. This approach results in the following formula for approximating the value I:

$$I = \int_a^b f(x)e^{ix}dx \approx \sum_{j=1}^{n} W_j f\left(x_j\right)e^{ix_j} ,$$

where the quadrature coefficients, $\{W_j\}$ and $\{x_j\}$, depend on the specific integration scheme being used (e.g., Gaussian, Lobatto, Simpson's rule, etc.), but can be computed independently of f(x). In order to reasonably approximate the integrand, $f(x)e^{ix}$, by a polynomial, the x_j must be located densely enough in the interval (a,b) to average at least two values for each cycle of e^{ix} (ref 1). If f(x) can be approximated by a fairly low order polynomial on (a,b), and if $|b - a| >> 2\pi$, the major part of the work in approximating the integrand involves approximating the oscillatory behavior of e^{ix}.

This difficulty just described is avoided in IFEIX by treating the factor e^{ix} as part of the integration operation, rather than as part of the integrand. Briefly, IFEIX is based on approximating f(x) by a polynomial and integrating this polynomial times e^{ix} over (a,b). This gives rise to approximation of the form

$$I = \int_a^b f(x)e^{ix}dx \approx \sum_{j=1}^{n} W_j f\left(x_j\right) ,$$

where $\{W_j\}$ and $\{x_j\}$, while different from before, can still be calculated independently of f(x).

This paper explains how the integral I can be represented so as to allow approximations of the form just outlined to be developed. The operation of the program IFEIX is explained and numerical examples are given. A program listing is appended.

2. MATHEMATICAL BACKGROUND.

We are considering evaluating integrands of the form

$$I = \int_a^b f(x)e^{ix}dx ,$$

where (1) $|b - a| >> 2\pi$, (2) f(x) is relatively slowly varying on (a,b) as compared to e^{ix}, and (3) as a practical necessity, f(x) is finite in the open interval (a,b), although it may be unbounded at the endpoints. The integral can be rewritten

$$I = e^{ia} \int_0^{(b-a)} f(x + a)e^{ix}dx .$$

We now introduce the notation

$$F(x) = f(x + a)$$

$$M = \left\lfloor \frac{(b - a)}{2\pi} \right\rfloor .$$

Then

$$I = e^{ia} \left[\int_0^{2\pi M} F(x)e^{ix}dx + \int_{2\pi M}^{(b-a)} F(x)e^{ix}dx \right] .$$

We would like to develop quadrature coefficients for the first of these integrals using e^{ix} as a weighting function. However, it is well known that suitable quadrature coefficients can only be developed for weighting functions that are everywhere nonnegative (ref 2). To this end, we add and subtract unity from the real and imaginary parts of e^{ix} in the first integral above to obtain

$$I = e^{ia} \left[\int_0^{2\pi M} F(x)(1 + i + e^{ix})dx - (1 + i) \int_0^{2\pi M} F(x)dx + \int_{2\pi M}^{(b-a)} F(x)e^{ix}dx \right] ,$$

or, defining the quantities

$$I_1 = \int_0^{2\pi M} (1 + \cos x)F(x)dx ,$$

$$I_2 = \int_0^{2\pi M} (1 + \sin x)F(x)dx,$$

$$I_3 = \int_0^{2\pi M} F(x)dx,$$

$$I_4 = \int_{2\pi M}^{(b-a)} F(x)e^{ix}dx ,$$

we have

$$I = e^{ia}\left[I_1 + iI_2 - (1 + i)I_3 + I_4\right] .$$

These four integrals can be evaluated separately and summed to provide an estimate for I. We note that the first two of these, I_1 and I_2, contain all of the oscillatory behavior of e^{ix} in the interval $(a + 2\pi M)$. The contribution of the remaining portion of the interval is contained in I_4. However, this latter integral represents less than one cycle of the exponential, and is easily integrated using existing integration programs. Integral I_3 is also relatively easy to evaluate using existing software, since it contains no oscillatory factor and, by assumption, f(x) varies relatively slowly. We are thus left with the problem of evaluating the integrals

$$I_1 = \int_0^{2\pi M} (1 + \cos x)F(x)dx,$$

$$I_2 = \int_0^{2\pi M} (1 + \sin x)F(x)dx.$$

Now, though, we are in a position to treat the oscillatory factors, (1 + cos x) and (1 + sin x), as weighting functions to be considered part of the integration operation, since they are everywhere nonnegative.

We now outline the procedure for incorporating the functions (1 + cos x) and (1 + sin x) into the integration operation. We wish to obtain approximations to I_1 and I_2 of the form

$$\int_0^{2\pi M} w(x)F(x)dx \approx \sum_{j-1}^{n} W_jF(x_j),$$

where w(x) = 1 + cos x or 1 + sin x for I_1 or I_2, respectively. The procedure to obtain $\{W_j\}$ and $\{x_j\}$ for each w(x) consists of three steps:

1. Choose a value for n, the order of integration. The approximation will then be exact for all F(x) equal to a polynomial up to degree 2n - 1.

2. Find the moments m_k, $0 \le k \le 2n - 1$, of the weighting function. That is, obtain the quantities

$$m_k = \int_0^{2\pi M} x^k w(x)dx \qquad 0 \le k \le 2n - 1.$$

3. Obtain the weights $\{W_j\}$ and abscissae $\{x_j\}$ as solutions to the 2n equations

$$\sum_{j=1}^{n} W_j x_j^k = m_k \qquad 0 \le k \le 2n - 1.$$

It is not immediately obvious how steps 2 and 3 should be performed. We discuss each of these in turn.

As to step 2, the most straightforward method of calculating the m_k for each w(x) is to develop a recursion relationship. This can be done by integrating $x^k w(x)$ by parts twice. For $w(x) = 1 + \cos x$, this yields

$$\left.\begin{array}{l} m_k = \dfrac{(2\pi M)^{k+1}}{k + 1} + 2k(2\pi M)^{k-1} - k(k - 1)m_{k-2} \\ \\ \text{while we obtain directly} \\ \\ m_0 = 2\pi M , \\ m_1 = \dfrac{(2\pi M)^2}{2} . \end{array}\right\} [w(x) = 1 + \cos x].$$

For $w(x) = 1 + \sin x$, we obtain the corresponding equation

$$\left.\begin{array}{l} m_0 = 2\pi M \\ m_1 = \dfrac{(2\pi M)^2}{2} - 2\ M \\ m_k = \dfrac{(2\pi M)^{k+1}}{k+1} - (2\pi M)^k + k(2\pi M)^{k-1} - k(k - 1)m_{k-2} \end{array}\right\} [w(x) = 1 + \sin x].$$

Thus, step 2 is easily performed.

Step 3, the solution to the 2n equations

$$\sum_{j=1}^{n} W_j x_j^k = m_k \qquad 0 \le k \le 2n - 1$$

is a fairly well-known problem, and several methods exist for solving these equations (ref 3 to 5). These methods are all based on calculating the $\{x_j\}$, either directly or indirectly, as the roots of the nth degree orthogonal polynomial from the system of orthogonal polynomials generated by w(x) on [0,2πM] (ref 2). The $\{W_j\}$ can then be obtained from the 2n equations of step 3, which are linear in the W_j.

As an example, we show in figures 1 and 2 the weights and abscissae for 1 + cos x and 1 + sin x for M = 6 and n = 10. We note in particular that for w(x) = 1 + sin x (figure 2), the weights and abscissae are not symmetric about the midpoint of the interval. This is a natural consequence of the fact that sin x is not an even function.

We have seen how the original integral

$$I = \int_a^b f(x)e^{ix}dx$$

can be broken down into an appropriate sum of four integrals, I_1 through I_4, and how each of these integrals can, in principle, be evaluated. The next section describes how these operations are carried out in IFEIX.

3. PROGRAM ORGANIZATION. If IFEIX were to carry out the evaluation of I as described in the last section, the major part of the time would be spent computing weights and abscissae for evaluating I_1 and I_2. However, without any additional constraints on the problem, it is impractical to tabulate $\{W_j\}$ and $\{x_j\}$ for all possible values of M. IFEIX resolves this problem by breaking the interval [0,2πM] into subintervals in a way that effectively reduces to ten the number of values of M that need to be considered.

Once this reduction has been performed, the integral over each subinterval is evaluated using an interval splitting and queueing algorithm in conjunction with an Aitken's δ^2 transformation. This combination of techniques has been used effectively in several other integration routines. It allows integral estimates to be obtained when the integrand has endpoint singularities. Also, it is adaptive in character, making the success of the integration very insensitive to the choice of order of approximation (the number of abscissae/weight pairs). We will now outline the basic subdivision procedure used by IFEIX. After that, the interval splitting and queueing algorithm will be explained.

In evaluating the integrals I_1 and I_2, where

$$I_i = \int_0^{2\pi M} w_i(x)F(x)dx,$$

in which $w_1(x) = 1 + \cos x$ and $w_2(x) = 1 + \sin x$, we wish to restrict our attention to only a few values of M. The subdivision procedure consists of four steps.

Step 1: Subdivide the interval $[0,2\pi M]$ into j subintervals $[0,2\pi M_1]$, $[2\pi M_1,2\pi M_2]$, . . . $[2\pi M_{j-1},2\pi M]$, where each interval is over no more than 640π. That is, for subinterval ℓ, $1 \leq \ell < j$,

$$M_\ell - M_{\ell-1} = 320 \qquad (\ell < j)$$

$$M - M_{j-1} \leq 320.$$

Step 2: Each subinterval ℓ, $1 \leq \ell \leq j$, is of the form

$$\int_{2\pi M_{\ell-1}}^{2\pi M_\ell} w(x)F(x)dx$$

and is treated as follows. If $M_\ell - M_{\ell-1} = 320$ or $M_\ell - M_{\ell-1} \leq 10$, proceed directly to step 3. If $10 < M_\ell - M_{\ell-1} < 320$, however, further subdivide interval ℓ into subintervals of length 10×2^k cycles for $0 \leq k \leq 4$, plus one more subinterval of less than 10 cycles. In other words represent $M_\ell - M_{\ell-1}$ as

$$M_\ell - M_{\ell-1} = 10 \times \sum_{k\in S} 2^k + M_R \qquad S \subset \{0,1,2,3,4\}$$

were, as indicated, S is a subset of the integers 0,1,2,3,4, and $M_R < 10$ is the remaining number of cycles in interval ℓ. Then perform step 3 for each subinterval of 10×2^k, or of M_R, cycles.

Step 3: The integral over each subinterval is of the form

$$\int_{2\pi M_{\ell-1}}^{2\pi M_\ell} w_i(x)F(x)dx = \int_0^{2\pi(M_\ell - M_{\ell-1})} w_i(x)F\left(x + 2\pi M_{\ell-1}\right)dx \quad ,$$

and is integrated using the interval splitting and queueing algorithm described below. We note that in this step we are always guaranteed that

$$M_\ell - M_{\ell-1} < 10$$

or

$$M_\ell - M_{\ell-1} = 10 \times 2^k \text{ for some k in } \{0,1,2,3,4,5\}.$$

Thus, integration coefficients are only needed for a few values of $M = M_\ell - M_{\ell-1}$, and our goal of restricting the number of cases for which integration coefficients are needed has been achieved.

Step 4: Add the results obtained in step 3 for each of the subintervals of step 2. This provides the final estimate for I_1 and I_2.

The only part of the organization of IFEIX which remains to be explained is the interval splitting and queueing algorithm and the δ^2

transformation as it is applied within that algorithm. This algorithm is basically identical to the one used in FOGIE, NL9, and CADRE (ref 6 to 8). We assume we are integrating a subinterval covering $[0,2\pi M]$. The algorithm itself is shown schematically in figure 3, which may aid in understanding the following description. The algorithm operates in cycles. Each cycle produces a new estimate of the integral over the subinterval. If $M > 5$ the first estimate is obtained by applying a 10-point integration formula to the integral over $[0,2\pi M]$. However if $M \leq 5$, the integration routine FOGIE is applied to the integral and the estimate returned by FOGIE is used as the final estimate. Assuming that $M > 5$, the algorithm continues as follows. The endpoints (0 and M) of the subinterval and the integral estimate are placed in a FIFO queue, which now holds only this one element. Then the following steps are repeated as indicated:

Step 1. Set ANSWER ← 0. Mark the last element of the queue as the end of a cycle. Set SUM ← 0.

Step 2. Remove the next queued interval. If there are no intervals queued, ANSWER is the value of the integral.

Step 3. If the interval is 5 or fewer cycles long, add the estimate to ANSWER. However, if it is more than 5 cycles long, form two subintervals by splitting the interval approximately in half, preserving endpoints that are integer multiples of 2π. (Note that if the queued interval is of length 10×2^k cycles, $k \geq 1$, then the two subintervals are of length $10 \times 2^{k-1}$ cycles.) If the length of the interval is 10 or fewer cycles, the subintervals are 5 or fewer (but at least 3) cycles long.

Step 4. Estimate the integral over each subinterval formed in step 3 using a 10-point formula. However, if a subinterval is 5 or fewer cycles long, estimate the integral using FOGIE.

Step 5. If the sum of the estimates obtained in step 4 are close enough (as defined by the convergence criteria) to the estimate retrieved from the queue in step 3, add the estimates from step 4 to ANSWER. If they are not sufficiently close, add the estimates to SUM and place each subinterval on the end of the queue.

Step 6. If the subinterval removed in step 3 was not marked as the end of a cycle, return to step 2. If it was so marked, then mark the subinterval last placed on the queue as the end of a cycle. (Note that at most one subinterval on the queue can be marked as the end of a cycle, so we can use a pointer into the queue as a marker.)

Step 7. At this point, ANSWER contains the sum of all subintervals for which a satisfactory estimate has been obtained, and SUM contains the sum of all subintervals still on the queue, for which the estimate may or may not be satisfactory. Thus ANSWER + SUM is a new estimate of the integral over the entire interval. Set ESTIM ← ANSWER + SUM and set SUM ← 0. If this estimate (ESTIM) is close enough to the previous estimate, then return ESTIM as the value of the integral. If ESTIM is not sufficiently close, then it forms a new estimate in a sequence of estimates. This sequence, if it contains at least four elements, can be accelerated using an Aitken's δ^2 transformation. Apply the transformation. If there are not four elements, or if the transformation does not converge, return to step 2. If the transformation does apply and results in an estimate Z which is judged the limit of the sequence, return Z as the estimate of the integral over the original interval.

This procedure is adaptive in nature, since it splits subintervals more finely only where better resolution is needed. Additionally, because of the convergence tests of step 7, troublesome sections of the interval which do not add significantly to the final answer need not be accurately integrated. Thus the procedure is fairly efficient. As has been shown by use in other routines, this procedure is also quite reliable and, as long as the integrand is continuous, is unlikely to return a spuriously convergent result.

The Aitken's δ^2 transformation is described in detail elsewhere (ref 9) and will only be outlined here. We assume we have a slowly converging sequence of estimates $Z_1, Z_2, \ldots, Z_n$. This sequence can be transformed into a new sequence $Z_3', Z_4', \ldots, Z_n'$ by the transformation

$$Z_k' = Z_k - \frac{\left(Z_k - Z_{k-1}\right)^2}{Z_k - 2Z_{k-1} + Z_{k-2}} \qquad k \geq 3.$$

The new sequence will be a more rapidly converging sequence than the original. In particular, if the errors of the original sequence are decreasing in geometric proportion, the new sequence will have all error removed.

The reason we require at least four estimates in step 7 before applying this transformation is that we need to have at least two elements in the transformed sequence in order to apply the convergence criteria to the transformed estimates. As long as there are at least four elements in the sequence of estimates, the δ^2 transformation can be applied repeatedly, as shown in figure 4. If at any step the difference between the last two estimates satisfies the error criteria, the last element of the sequence can be taken as the limit of the sequence.

4. RESULTS. IFEIX has been applied to various integrals and has performed reliably and efficiently. Two of the more dramatic results are reported here.

The first problem was to evaluate the integral

$$\int_1^{5000} x^3 e^{ix} dx \approx (-12.3484175 - i\ 1.94076436) \times 10^{10}$$

IFEIX was used with an error criterion requesting a relative error of 10^{-6}. The answer returned by IFEIX was

$$\text{ANSWER} = (-12.3484\underline{275} - i\ 1.9407\underline{7072}) \times 10^{10},$$

and was obtained with 872 function evaluations. The portions of ANSWER that differ from the true value of the integral are underlined. For comparison, this integral was attempted using FOGIE, modified to handle complex-valued integrals. A convergent answer could not be obtained within the limit of 100,000 function evaluations.

The second example of the use of IFEIX is taken from an actual application. Integrals of the form

$$\int_A^B \frac{e^{ix}dx}{x\sqrt{\left(x^2 - A^2\right)\left(B^2 - x^2\right)}}$$

arise in certain physical optics problems (ref 10). Note that singularities appear at both endpoints of the interval of integration. Both IFEIX and FOGIE were used to evaluate about 200 integrals of this form for various large, positive values of A and B for which

$$1 \leq \frac{B - A}{2\pi} \leq 40.$$

The resulting comparison of run times revealed that

$$\frac{\text{CPU Time for IFEIX}}{\text{CPU Time for FOGIE}} \simeq \frac{1}{70},$$

a substantial improvement in performance.

5. CONCLUSIONS. We have outlined the operation of IFEIX, a FORTRAN routine for evaluating integrals with a factor of e^{ix} in the integral. Detailed instructions for the use of IFEIX are given in the comment lines appearing at the beginning of the program. (The source code for the program is given in the Appendix.)

IFEIX has been shown to be a reliable routine that can provide vast improvements in efficiency over existing routines in problems for which it was designed. We strongly recommend the use of IFEIX when an oscillatory factor e^{ix} (or cos x or sin x) is the most rapidly varying factor of the integrand.

LITERATURE CITED.

1. Tukey, J. W., "The Estimation of (Power) Spectra and Related Quantities," in On Numerical Approximation, R. E. Langer, Ed., pp. 389-411, Madison U. Press, Madison, Wisconsin (1959).

2. Beckmann, P., Orthogonal Polynomials for Engineers and Scientists, The Golem Press, Boulder, Colorado (1973).

3. Hopp, T., QINTEG - A Program for Computing Gaussian Quadrature Coefficients, Harry Diamond Laboratories, HDL-TR-1890 (1979).

4. Golub, G. H., and Welch, J. H., "Calculation of Gauss Quadrature Rules," Comput. Sci. Dept. Tech. Rep. No. CS 81, Stanford University, Calif. (1967).

5. Gautschi, W., "Construction of Gauss-Christoffel Quadrature Formulas," Math. of Comput., V. 22 (1968), pp. 251-270.

6. Hausner, A., and Hutchinson, J. D., "FOGIE: An Adaptive Code for Numerical Integrals Using Gaussian Quadrature," ARO Report 75-3, Proc. 1975 Army Num. Analysis and Computer Conf., pp 139-177.

7. Hausner, A., "NL9 - An Adaptive Routine for Numerical Quadrature," ARO Report 77-3, Proc. 1977 Army Num. Analysis and Computers Conf., pp. 367-410.

8. de Boor, C., "CADRE: An Algorithm for Numerical Quadrature," Mathematical Software, John R. Rice, Ed., Academic Press, N.Y. (1971), pp. 417-449.

9. Shanks, D., "Non-linear Transformation of Divergent and Slowly Convergent Sequences," Journal of Math. Physics, 34 (1955), pp. 1-42.

10. Hopp, T., "Numerical Calculation of the Mutual Intensity of Coherent Light Reflected from a Goodman Surface," Harry Diamond Laboratories, to be published (1979).

FIGURE 1. INTEGRATION WEIGHTS AND ABSCISSAE FOR W(X)=1+COS(X) - 6 CYCLES. INTERVAL = (0,12*PI)

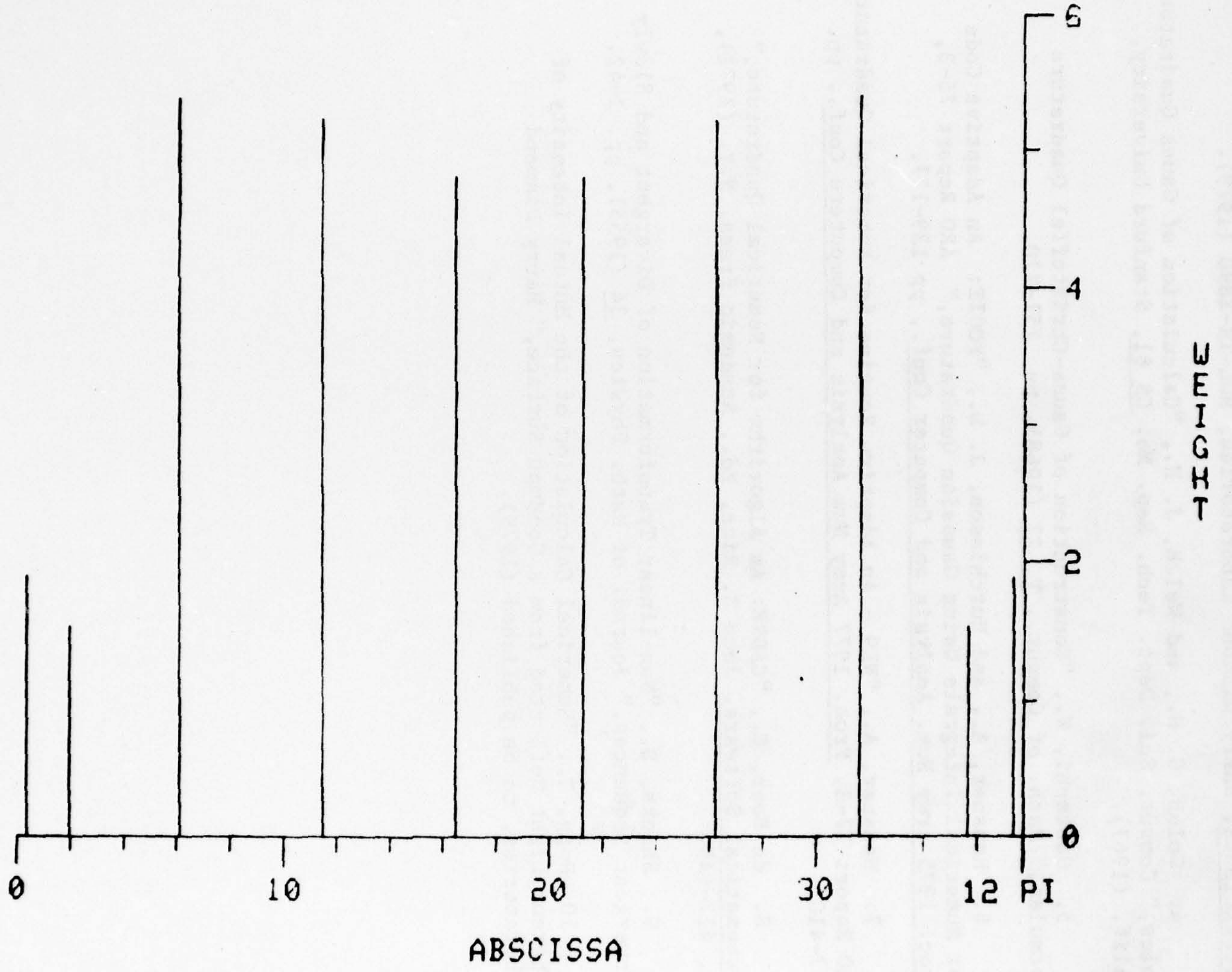


FIGURE 2. INTEGRATION WEIGHTS AND ABSCISSAE FOR W(X)=1+SIN(X) - 6 CYCLES. INTERVAL = (0,12*PI)

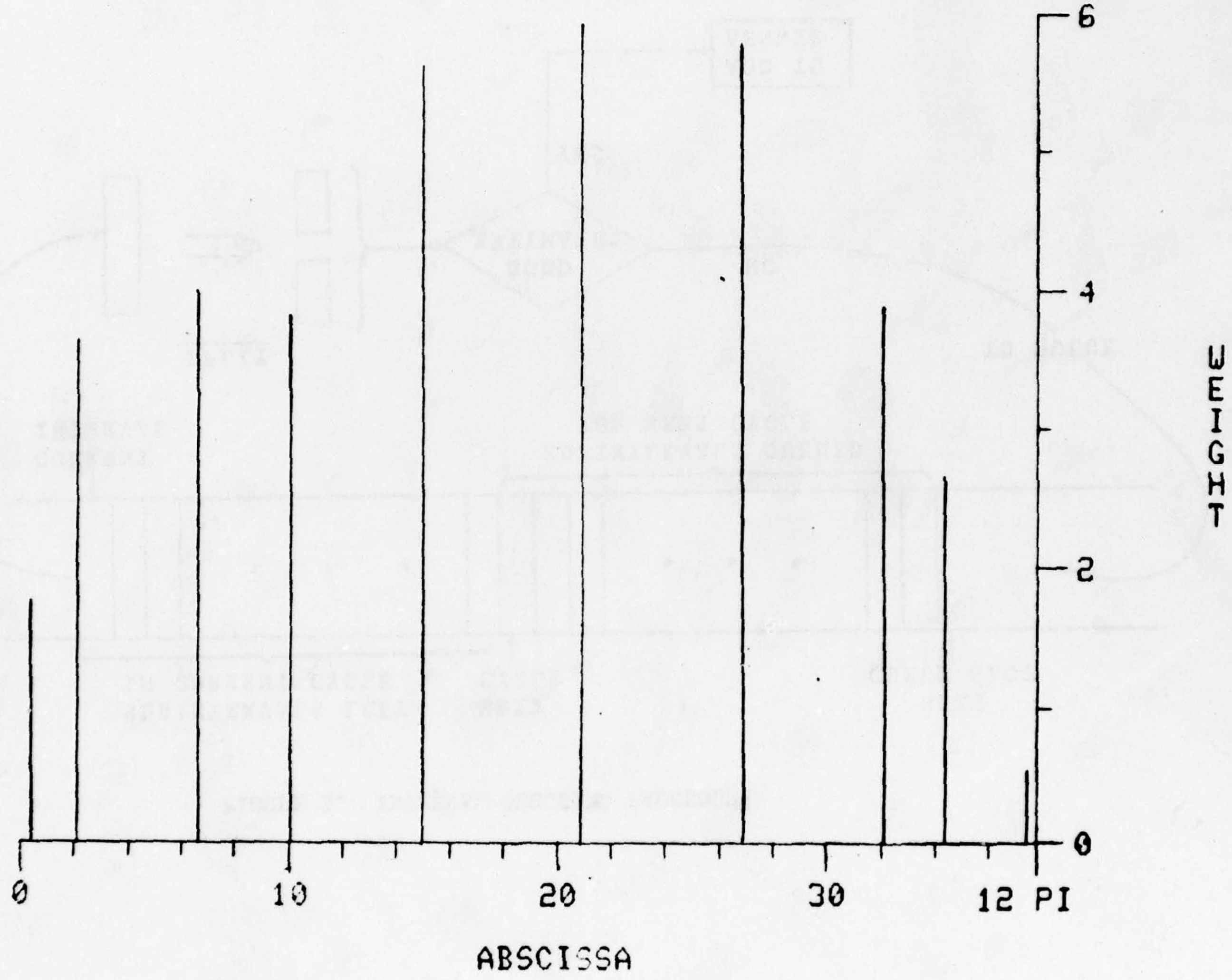


FIGURE 3. INTERVAL QUEUEING PROCEDURE

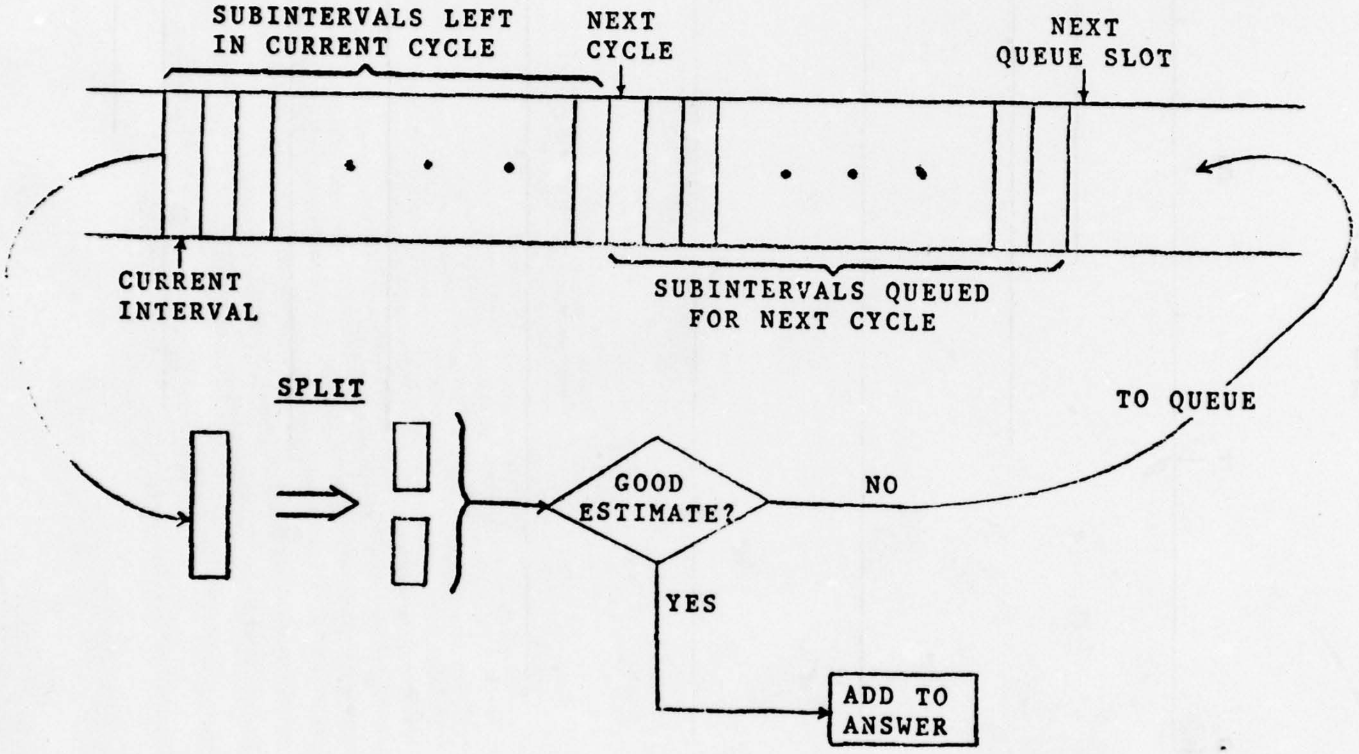
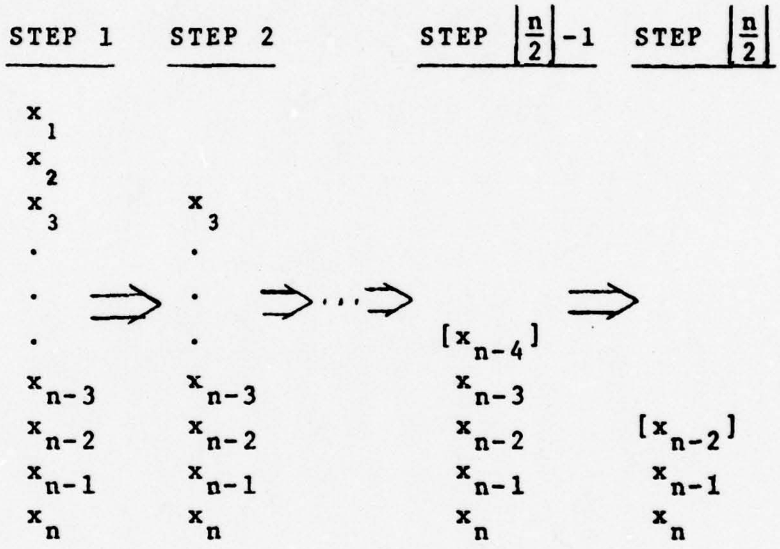


FIGURE 4. REPEATED APPLICATION OF AITKEN'S δ^2 TRANSFORMATION



APPENDIX

The following is a listing of the FORTRAN code for IFEIX, along with the principal subroutines it uses. The comment cards prefacing the principal subroutire, IFEIX, provide all information necessary for its use.

```
      SUBROUTINE IFEIX(ANSR,ANSI,A,B,LIMIT,DACC,FOFX,IFLAG,ICODE,IERR)  FEIX0010
C   PURPOSE                                                             FEIX0020
C     THIS SUBROUTINE INTEGRATES F(X)*EXP(IX) FROM A TO B.  THE INTER-  FEIX0030
C     VAL (A,B) IS DIVIDED INTO SECTIONS OF NO MORE THAN 320 CYCLES OF  FEIX0040
C     THE EXPONENTIAL, AND EACH SECTION IS INTEGRATED SEPARATELY BY     FEIX0050
C     SUBROUTINE 'SPLIT', WHICH USES A GENERAL GAUSSIAN INTEGRATION     FEIX0060
C     SCHEME WITH TABLE LOOK-UP OF INTEGRATION ABSCISSAE AND WEIGHTS.   FEIX0070
C                                                                       FEIX0080
C   ARGUMENTS                                                           FEIX0090
C     ANSR  - REAL*8 OUTPUT VARIABLE.  ON RETURN, THIS CONTAINS THE     FEIX0100
C             REAL PART OF THE INTEGRAL OF F(X)*EXP(IX) FROM A TO B IF  FEIX0110
C             REQUESTED BY ICODE (SEE BELOW).                           FEIX0120
C     ANSI  - REAL*8 OUTPUT VARIABLE.  ON RETURN, THIS CONTAINS THE     FEIX0130
C             IMAGINARY PART OF THE INTEGRAL OF F(X)*EXP(IX) FROM A     FEIX0140
C             TO B IF REQUESTED BY ICODE (SEE BELOW).                   FEIX0150
C     A     - REAL*8 INPUT SCALAR.  THE LOWER LIMIT OF INTEGRATION.     FEIX0160
C     B     - REAL*8 INPUT SCALAR.  THE UPPER LIMIT OF INTEGRATION.     FEIX0170
C     LIMIT - INTEGER*4 INPUT SCALAR.  MAXIMUM NUMBER OF FUNCTION       FEIX0180
C             EVALUATIONS TO BE ALLOWED FOR EACH SECTION OF AT MOST 320 FEIX0190
C             CYCLES.                                                   FEIX0200
C     DACC  - REAL*8 INPUT SCALAR.  THE ERROR PARAMETER.  THE PROGRAM   FEIX0210
C             RETURNS WHEN IT ESTIMATES THAT ABS(ANS-TRUE ANSWER)       FEIX0220
C             IS LESS THAN OR EQUAL TO DACC (OR DACC*TRUE ANSWER        FEIX0230
C             IF A RELATIVE ERROR IS SPECIFIED - SEE IERR BELOW).       FEIX0240
C     FOFX  - SUBROUTINE NAME.  THIS NAME MUST BE DECLARED EXTERNAL     FEIX0250
C             IN THE CALLING PROGRAM.  SUBROUTINE FOFX COMPUTES F(X)    FEIX0260
C             WHEN CALLED WITH THE SEQUENCE                             FEIX0270
C                  CALL FOFX(X,F)                                       FEIX0280
C             WHERE X AND F ARE BOTH REAL*8 SCALARS.                    FEIX0290
C     IFLAG - INTEGER*4 INPUT SCALAR.  THIS VARIABLE INDICATES WHETHER  FEIX0300
C             THE PROGRAM CAN ASSUME THAT 'A' IS AN INTEGER MULTIPLE    FEIX0310
C             OF PI/2 AS FOLLOWS:                                       FEIX0320
C                IFLAG=1 - 'A' IS AN ODD INTEGER MULTIPLE OF PI/2.      FEIX0330
C                IFLAG=2 - 'A' IS AN EVEN INTEGER MULTIPLE OF PI/2.     FEIX0340
C             ANY OTHER VALUE OF IFLAG INDICATES NEITHER CONDITION      FEIX0350
C             CAN BE ASSUMED.                                           FEIX0360
C     ICODE - INTEGER*4 INPUT SCALAR.  THIS VARIABLE INDICATES WHETHER  FEIX0370
C             THE REAL, IMAGINARY, OR BOTH PARTS OF THE INTEGRAL ARE    FEIX0380
C             TO BE COMPUTED AS FOLLOWS:                                FEIX0390
C                ICODE=1 - COMPUTE THE REAL PART ONLY.                  FEIX0400
C                ICODE=2 - COMPUTE THE IMAGINARY PART ONLY.             FEIX0410
C             ANY OTHER VALUE OF ICODE CAUSES BOTH THE REAL AND         FEIX0420
C             IMAGINARY PARTS OF THE INTEGRAL TO BE COMPUTED.           FEIX0430
C     IERR  - INTEGER*4 INPUT AND OUTPUT SCALAR.                        FEIX0440
C             INPUT - ERROR TOLERANCE CODE.  IF:                        FEIX0450
C                IERR=0 - DACC IS AN ABSOLUTE ERROR TOLERANCE.          FEIX0460
C                IERR=1 - DACC IS A RELATIVE ERROR TOLERANCE.           FEIX0470
C                  ANY OTHER VALUE OF IERR IS TREATED AS IERR=1.        FEIX0480
C             OUTPUT - RETURN ERROR CODE.  IF:                          FEIX0490
C                IERR=0 - AN ESTIMATE HAS BEEN OBTAINED.                FEIX0500
C                IERR=1 - AN UNRELIABLE ESTIMATE HAS BEEN OBTAINED.     FEIX0510
C                         THE LIMIT ON THE NUMBER OF FUNCTION EVALUA-   FEIX0520
C                         TIONS FORCED AN UNRELIABLE ESTIMATE OF THE    FEIX0530
C                         INTEGRAL TO BE USED FOR SOME SUBINTERVAL.     FEIX0540
C                IERR=2 - NO ESTIMATE POSSIBLE.  THE QUEUE FOR INTERVAL FEIX0550
C                         SPLITTING WAS FILLED FOR SOME SUBINTERVAL.    FEIX0560
C                         THE REAL AND IMAGINARY PARTS OF THE ANSWER    FEIX0570
C                         ARE SET TO ZERO.                              FEIX0580
C                                                                       FEIX0590
```

```
C     PROGRAM NOTES                                                     FFIX0600
C       THE FIRST TWO VARIABLES OF COMMON BLOCK 'CMSPLT' ARE OF INTEREST. FFIX0610
C       THEY ARE 'BASE', A REAL*8 VARIABLE EQUAL TO 'A' OF THE ARGUMENT  FFIX0620
C       LIST, AND 'LTOTAL', AN INTEGER*4 VARIABLE FOR THE TOTAL NUMBER OF FFIX0630
C       FUNCTION EVALUATIONS USED TO OBTAIN THE ANSWER. THIS MAY BE LAR- FFIX0640
C       GER THAN 'LIMIT' OF THE ARGUMENT LIST, SINCE 'LIMIT' APPLIES ONLY FFIX0650
C       TO THE INTEGRATION OF EACH SECTION OF 320 OR FEWER CYCLES, OR    FFIX0660
C       TO THE INTEGRATION OF F(X) BY FOGIE WITHOUT THE EXPONENTIAL.     FFIX0670
C       THESE VARIABLES MAY BE ACCESSED AFTER FFIX RETURNS BY            FFIX0680
C       INCLUDING 'CMSPLT' IN A COMMON DECLARATION IN THE CALLING        FFIX0690
C       PROGRAM.                                                         FFIX0700
C                                                                        FFIX0710
C     SUBROUTINES USED (* INDICATES NOT INCLUDED IN THIS MODULE)         FFIX0720
C      *FOGIE  (HDL PAGLOAD LIBRARY)                                     FFIX0730
C       CMIF   (COMMON AREA)                                             FFIX0740
C       CMSPLT (COMMON AREA)                                             FFIX0750
C       CMWGT  (COMMON AREA)                                             FFIX0760
C       SPLIT                                                            FFIX0770
C       FCOSXA                                                           FFIX0780
C       FSINXA                                                           FFIX0790
C       FCOSX                                                            FFIX0800
C       FSINX                                                            FFIX0810
C       FLINK                                                            FFIX0820
C       COSWT (COS6,COS7,COS8,COS9,COS10,COS20,COS40,COS80,COS160,COS320) FFIX0830
C       SINWT (SIN6,SIN7,SIN8,SIN9,SIN10,SIN20,SIN40,SIN80,SIN160,SIN320) FFIX0840
C                                                                        FFIX0850
C     AUTHOR                                                             FFIX0860
C       TED HOPP, HDL.                                                   FFIX0870
C                                                                        FFIX0880
        COMMON /CMSPLT/BASE,LTOTAL,L,ACC,LIM/CMIF/FOFXCM                 FFIX0890
        DIMENSION INCHS(5)                                               FFIX0900
        REAL*8 TWOPI,S1,S2,S3,S4,A,B,C,ACC,DACC,BASE,ANSR,ANSI           FFIX0910
        EXTERNAL FCOSXA,FSINXA,FCOSX,FSINX,COSWT,SINWT                   FFIX0920
        DATA INCHS/160,80,40,20,10/,TWOPI/6.283185307179586476925287D0/  FFIX0930
        LIM=LIMIT                                                        FFIX0940
        ACC=.1D0*DACC                                                    FFIX0950
        IACC=1                                                           FFIX0960
        IF (IERR.EQ.0) IACC=0                                            FFIX0970
        IERR=0                                                           FFIX0980
        BASE=A                                                           FFIX0990
        M=(B-A)/TWOPI                                                    FFIX1000
        C=TWOPI*DFLOAT(M)                                                FFIX1010
        N=M/320                                                          FFIX1020
        S2=0.D0                                                          FFIX1030
        S3=0.D0                                                          FFIX1040
        IBASE=0                                                          FFIX1050
        FOFXCM=FOFX                                                      FFIX1060
C                                                                        FFIX1070
C     N IS THE NUMBER OF SECTIONS OF 320 CYCLES.                         FFIX1080
C                                                                        FFIX1090
        IF (N.EQ.0) GO TO 40                                             FFIX1100
C                                                                        FFIX1110
C     EVALUATE EACH SECTION OF 320 CYCLES SEPARATELY                     FFIX1120
C                                                                        FFIX1130
        DO 30 I=1,N                                                      FFIX1140
        IF (ICODE.EQ.2.AND.IFLAG.EQ.1) GO TO 10                          FFIX1150
        CALL SPLIT(FCOSXA,COSWT,IBASE,IBASE+320,S1,IACC,IER)             FFIX1160
        IF (IER.LT.-LIM) GO TO 120                                       FFIX1170
        IF (IER.LT.0) IERR=1                                             FFIX1180
```

```
      S2=S2+S1
      LTOTAL=LTOTAL+L
      IF (ICODE.EQ.1.AND.IFLAG.EQ.2) GO TO 20
   10 CALL SPLIT(FSINXA,SINWT,IBASE,IBASE+320,S1,IACC,IER)
      IF (IER.LT.-LIM) GO TO 120
      IF (IER.LT.0) IERR=1
      S3=S3+S1
      LTOTAL=LTOTAL+L
   20 IBASE=IBASE+320
   30 CONTINUE
C
C  M IS THE NUMBER OF COMPLETE CYCLES IN THE OVERALL INTEGRATION.
C  IBASE IS THE NUMBER OF COMPLETE CYCLES ALREADY INTEGRATED.
C
   40 IF (M.EQ.IBASE) GO TO 90
C
C  EVALUATE REMAINING SECTION (OF LESS THAN 320 CYCLES) IN
C  BLOCKS OF 10*2**K CYCLES UNTIL THERE ARE 10 OR FEWER CYCLES.
C
      DO 70 I=1,5
      IF (M.LT.IBASE+INCRS(I)) GO TO 70
      IF (ICODE.EQ.2.AND.IFLAG.EQ.1) GO TO 50
      CALL SPLIT(FCOSXA,COSWT,IBASE,IBASE+INCRS(I),S1,IACC,IER)
      IF (IER.LT.-LIM) GO TO 120
      IF (IER.LT.0) IERR=1
      S2=S2+S1
      LTOTAL=LTOTAL+L
      IF (ICODE.EQ.1.AND.IFLAG.EQ.2) GO TO 60
   50 CALL SPLIT(FSINXA,SINWT,IBASE,IBASE+INCRS(I),S1,IACC,IER)
      IF (IER.LT.-LIM) GO TO 120
      IF (IER.LT.0) IERR=1
      S3=S3+S1
      LTOTAL=LTOTAL+L
   60 IBASE=IBASE+INCRS(I)
      GO TO 40
   70 CONTINUE
C
C  EVALUATE LAST SECTION OF FEWER THAN 10 CYCLES.
C
      IF (ICODE.EQ.2.AND.IFLAG.EQ.1) GO TO 80
      CALL SPLIT(FCOSXA,COSWT,IBASE,M,S1,IACC,IER)
      IF (IER.LT.-LIM) GO TO 120
      IF (IER.LT.0) IERR=1
      S2=S2+S1
      LTOTAL=LTOTAL+L
      IF (ICODE.EQ.1.AND.IFLAG.EQ.2) GO TO 90
   80 CALL SPLIT(FSINXA,SINWT,IBASE,M,S1,IACC,IER)
      IF (IER.LT.-LIM) GO TO 120
      IF (IER.LT.0) IERR=1
      S3=S3+S1
      LTOTAL=LTOTAL+L
C
C  EVALUATE INTEGRAL OF F(X) FROM A TO A+2*PI*M
C
   90 CALL FOGIE(FOFX,0,1,A,A+C,SNGL(ACC),IACC,LIM,S4,IER)
      IF (IER.LT.-LIM) GO TO 120
      IF (IER.LT.0) IERR=1
      LTOTAL=LTOTAL+IABS(IER)
C
```

```
C     EVALUATE LAST SECTION OF LESS THAN ONE CYCLE.                     FEIX1780
C                                                                       FEIX1790
      ANSR=0.D0                                                         FEIX1800
      ANSI=0.D0                                                         FEIX1810
      IF (ICODE.EQ.2.AND.IFLAG.EQ.1) GO TO 100                          FEIX1820
      CALL POGIE(FCOSX,0,1,A+C,B,SNGL(ACC),IACC,LIM,ANSR,IER)           FEIX1830
      IF (IER.LT.-LIM) GO TO 120                                        FEIX1840
      IF (IER.LT.0) IERR=1                                              FEIX1850
      LTOTAL=LTOTAL+IABS(IER)                                           FEIX1860
      IF (ICODE.EQ.1.AND.IFLAG.EQ.2) GO TO 110                          FEIX1870
100   CALL POGIE(FSINX,0,1,A+C,B,SNGL(ACC),IACC,LIM,ANSI,IER)           FEIX1880
      IF (IER.LT.-LIM) GO TO 120                                        FEIX1890
      IF (IER.LT.0) IERR=1                                              FEIX1900
      LTOTAL=LTOTAL+IABS(IER)                                           FEIX1910
C                                                                       FEIX1920
C     EVALUATE ANSWER AS THE SUM OF THE SECTION INTEGRALS               FEIX1930
C     LESS THE INTEGRAL OF (1+SIGN(-1))*F(X).                           FEIX1940
C                                                                       FEIX1950
      IF (ICODE.NE.1.OR.IFLAG.NE.2)                                     FEIX1960
     *   ANSI=ANSI+DCOS(A)*(S3-S4)+DSIN(A)*(S2-S4)                      FEIX1970
110   IF (ICODE.NE.2.OR.IFLAG.NE.1)                                     FEIX1980
     *   ANSR=ANSR+DCOS(A)*(S2-S4)-DSIN(A)*(S3-S4)                      FEIX1990
      RETURN                                                            FEIX2000
C                                                                       FEIX2010
C     ERROR CODES INDICATE QUEUE WAS FILLED.                            FEIX2020
C                                                                       FEIX2030
120   IERR=2                                                            FEIX2040
      ANSR=0.D0                                                         FEIX2050
      ANSI=0.D0                                                         FEIX2060
      LTOTAL=LTOTAL-IER-LIM                                             FEIX2070
      RETURN                                                            FEIX2080
      END                                                               FEIX2090
      SUBROUTINE SPLIT(FCT1,WEIGHT,M1,M2,ANS,IACC,IER)
C   PURPOSE
C     THIS SUBROUTINE INTEGRATES WEIGHT(X)*FCT(X+BASE) FROM 2*PI*M1
C     TO 2*PI*M2 BY INTERVAL HALVING.  A TEN POINT GAUSSIAN
C     INTEGRATION FORMULA WITH A WEIGHTING FUNCTION 'WEIGHT'
C     IS USED THROUGHOUT.  SUCCESSIVE ESTIMATES ARE COMPUTED
C     AS SUMS OF FCT(X(I)) OVER, FIRST, THE ENTIRE INTERVAL,
C     AND THEN OVER 2,4,8,... SUBINTERVALS SUCCESSIVELY.  ESTIMATES
C     ARE PLACED IN THE ARRAY Z, WHICH IS USED BY SUBROUTINE
C     'ERROR' TO DETERMINE IF ANY ESTIMATE OR EXTRAPOLATED VALUE
C     FOR THE INTEGRAL IS SATISFACTORY.
C
C   ARGUMENTS
C     FCT1   - NAME OF A SUBROUTINE TO COMPUTE WEIGHT(X)*FCT(X)
C     WEIGHT - NAME OF A SUBROUTINE TO COMPUTE ABSCISSAE AND
C              WEIGHTS FOR INTEGRATION WITH RESPECT TO A WEIGHTING
C              FUNCTION.
C     M1     - LOWER LIMIT OF INTEGRATION (WHEN MULTIPLIED BY 2*PI).
C     M2     - UPPER LIMIT OF INTEGRATION (WHEN MULTIPLIED BY 2*PI).
C     ANS    - REAL*8 OUTPUT SCALAR.  THE ESTIMATE OF THE INTEGRAL.
C     IACC   - INTEGER*4 INPUT SCALAR.  ERROR TOLERANCE CODE.  IF
C                        IACC=0 - ACC IS AN ABSOLUTE ERROR TOLERANCE,
C                        IACC=1 - ACC IS A RELATIVE ERROR TOLERANCE.
C     IER    - INTEGER*4 OUTPUT SCALAR.  EXECUTION ERROR CODE.  IF
C                 IER>0 - THE COMPUTATION WAS SUCCESSFUL.  IER IS
C                         THE TOTAL NUMBER OF FUNCTION EVALUATIONS.
C            -LIM<IER<0 - THE COMPUTATION WAS ABORTED BECAUSE
C                         OF EXCESSIVE FUNCTION EVALUATIONS.  BEST
C                         ESTIMATE IS RETURNED.
C               IER<-LIM - THE COMPUTATION WAS ABORTED BECAUSE
C                         THE QUEUE WAS FILLED.  NO ESTIMATE GIVEN.
C
C   PROGRAM NOTES
C     VARIABLES BASE, LTOTAL, L, ACC, AND LIM APPEAR IN COMMON SECTION
C     'CMSPLT'.  LIM AND ACC MUST BE SET BY THE CALLING PROGRAM BEFORE
C     THE CALL TO SPLIT.  THE VARIABLES HAVE THE FOLLOWING MEANING:
C        BASE   - REAL*8 SCALAR.  THE DISPLACEMENT OF THE WEIGHTING
C                 FUNCTION DOMAIN ORIGIN FROM THE INTEGRAND DOMAIN
C                 ORIGIN.  SEE 'PURPOSE'.
C        LTOTAL - INTEGER*4 SCALAR.  THE TOTAL NUMBER OF FUNCTION
C                 EVALUATIONS TO DATE.
C        L      - INTEGER*4 SCALAR.  THE TOTAL NUMBER OF FUNCTION
C                 EVALUATIONS TO DATE.
C        ACC    - REAL*8 SCALAR.  THE ERROR TOLERANCE.
C        LIM    - INTEGER*4 SCALAR.  THE MAXIMUM NUMBER OF FUNCTION
C                 EVALUATIONS TO BE ALLOWED.
C
      COMMON /CMSPLT/BASE,LTOTAL,L,ACC,LIM/CMIF/FCT
      REAL*8 ANS,ACC,ERR2,LEFT,RIGHT,ESTIM(100),ERR,EST,ENTRYS,SUM
      REAL*8 BASE,Z(20)
      DIMENSION MEND1(100),MEND2(100)
      INTEGER FILEND,CYCLE,POINT
      EXTERNAL WEIGHT,FCT1
      DATA LEN/100/
C
C   INITIALIZE QUEUE AND POINTERS
C
      L=0
      I=1
      ERR2=DMAX1(1.D-15,ACC*1.D-2)
      MEND1(2)=M1
      MEND2(2)=M2
      CALL INIT
      CALL EVAL(M1,M2,WEIGHT,FCT,FCT1,ESTIM(2),IER)
      IF (IER.LT.0) GO TO 90
      Z(1)=ESTIM(2)
      FILEND=2
      POINT=3
      CYCLE=POINT
      SUM=0.D0
C
C     THIS BEGINS A NEW CYCLE.  'ENTRYS' WILL ACCUMULATE
C     THE ESTIMATES OF THE INTERVAL INTEGRALS FOR INTERVALS
C     SPLIT AND REPLACED IN THE QUEUE ON THIS CYCLE.
C
10    ENTRYS=0.D0
C
C     SPLIT INTERVAL AND EVALUATE INTEGRAL OF EACH HALF.  REMEMBER TO
C     CHECK IF 'LIM' WAS ABOUT TO BE EXCEEDED.
C
20    MA=MEND1(FILEND)
      MB=MEND2(FILEND)
C
C     DO NOT SPLIT INTERVALS THAT ARE SHORTER THAN 6 CYCLES.
C
      IF (MB-MA.GT.5) GO TO 30
      EST=ESTIM(FILEND)
      IER=L
      GO TO 40
30    MC=(MA+MB)/2
      CALL EVAL(MA,MC,WEIGHT,FCT,FCT1,LEFT,IER)
      IF (IER.LT.0) GO TO 90
      CALL EVAL(MC,MB,WEIGHT,FCT,FCT1,RIGHT,IER)
      IF (IER.LT.0) GO TO 90
C
C     NEW ESTIMATE IS THE SUM OF THE INTEGRALS OF THE TWO HALF-INTERVALS
C
      EST=LEFT+RIGHT
40    ERR=ESTIM(FILEND)-EST
C
C     REMOVE OLD INTERVAL FROM QUEUE.  COMPARE OLD ESTIMATE TO NEW.
C
      FILEND=MOD(FILEND,LEN)+1
      IF (ERR.EQ.0.D0) GO TO 60
      IF (EST.EQ.0.D0) GO TO 50
      IF (IACC.EQ.1.OR.DABS(EST).LT.1.D0) ERR=ERR/EST
      IF (DABS(ERR).LE.ERR2) GO TO 60
C
C     SUCCESSIVE ESTIMATES NOT CLOSE ENOUGH - QUEUE SUBINTERVALS
C
50    POINT=MOD(POINT,LEN)
      IF (FILEND-1.EQ.POINT) GO TO 100
      MEND1(POINT)=MA
      MEND2(POINT)=MC
      ESTIM(POINT)=LEFT
      POINT=POINT+1
      WEND1(POINT)=WC                                                   SPLT1170
      WEND2(POINT)=WB                                                   SPLT1180
      ESTIM(POINT)=RIGHT                                                SPLT1190
      POINT=POINT+1                                                     SPLT1200
      ENTRYS=ENTRYS+EST                                                 SPLT1210
      GO TO 70                                                          SPLT1220
C                                                                       SPLT1230
C   INTERVAL ESTIMATE IS SATISFACTORY.  ADD TO CUMULATIVE SUM           SPLT1240
C                                                                       SPLT1250
60    SUM=SUM+EST                                                       SPLT1260
      IF (FILEND.NE.POINT) GO TO 70                                     SPLT1270
C                                                                       SPLT1280
C   QUEUE EMPTY - INTEGRAL OBTAINED.                                    SPLT1290
C                                                                       SPLT1300
      ANS=SUM                                                           SPLT1310
      RETURN                                                            SPLT1320
70    IF (FILEND.NE.CYCLE) GO TO 20                                     SPLT1330
C                                                                       SPLT1340
C   A CYCLE IS COMPLETE.  PRODUCE A NEW OVERALL ESTIMATE AND            SPLT1350
C   CHECK IF TOTAL INTEGRAL ESTIMATES ARE CONVERGING ENOUGH TO STOP.    SPLT1360
C                                                                       SPLT1370
      I=I+1                                                             SPLT1380
      Z(I)=SUM+ENTRYS                                                   SPLT1390
      CALL ERRORA(I,Z,IACC,IER)                                         SPLT1400
      IF (IER.LT.0) GO TO 80                                            SPLT1410
C                                                                       SPLT1420
C   ERROR TEST HAS PASSED.  ANSWER IS GOOD UNLESS IT IS TOO             SPLT1430
C   SMALL AND ABSOLUTE ERROR TOLERANCE IS BEING USED.                   SPLT1440
C                                                                       SPLT1450
      IF (I.NE.2.OR.IACC.EQ.1.OR.DABS(Z(2)).GT.ACC*1.D1) GO TO 90       SPLT1460
C                                                                       SPLT1470
C   ERROR TEST FAILED.  MARK THE START OF ANOTHER CYCLE.                SPLT1480
C                                                                       SPLT1490
80    CYCLE=MOD(POINT,LEN)                                              SPLT1500
      GO TO 10                                                          SPLT1510
C                                                                       SPLT1520
C   ERROR TEST PASSED.  SET ANSWER AND EXIT SUBROUTINE.                 SPLT1530
C                                                                       SPLT1540
90    ANS=Z(I)                                                          SPLT1550
      RETURN                                                            SPLT1560
C                                                                       SPLT1570
C   THE QUEUE IS FILLED.  SET ERROR CODE AND EXIT.                      SPLT1580
C                                                                       SPLT1590
100   IER=-L-LIM                                                        SPLT1600
      RETURN                                                            SPLT1610
      END                                                               SPLT1620
      SUBROUTINE ERRORA(J,Z,IACC,IER)
C     PURPOSE
C       THIS SUBROUTINE PERFORMS SUCCESSIVE E-SUB-1 TRANSFORMATIONS ON
C       THE Z ARRAY AND DETERMINES IF AN ESTIMATE IS SATISFACTORY.
C
C     ARGUMENTS
C       J    - INTEGER*4 INPUT SCALAR.  THE NUMBER OF ESTIMATES CONTAINED
C              IN Z ON INPUT.  ON OUTPUT, THIS NUMBER MAY BE CHANGED.
C       Z    - REAL*8 INPUT ARRAY.  THE SUCCESSIVE ESTIMATES OF THE
C              INTEGRAL, FROM Z(1) TO Z(J).
C       IACC - INTEGER*4 INPUT SCALAR.  ERROR TOLERANCE CONTROL.  IF
C                     IACC=0 - ACC IS AN ABSOLUTE ERROR TOLERANCE.
C                     IACC=1 - ACC IS A RELATIVE ERROR TOLERANCE.
C       IER  - INTEGER*4 OUTPUT SCALAR.  EXECUTION ERROR CODE.  IF
C                     IER>0 - Z(J) IS THE FINAL ESTIMATE
C                     IER<0 - NO ESTIMATE IS GOOD ENOUGH.
C              THE MAGNITUDE OF IER WILL BE THE NUMBER OF FUNCTION
C              EVALUATIONS USED TO DATE, AS PASSED IN COMMON IN 'L'
C
      COMMON /CMSPLT/BASE,LTOTAL,L,ACC,LIM
      REAL*8 D,E,Z(1),W(20),ACC,BASE
C
C     CHECK IF EXTRAPOLATION CAN BE PERFORMED.
C
      IF (J.EQ.2) GO TO 10
      IF (DABS(Z(J)-Z(J-1)).LT.DABS(Z(J-1)-Z(J-2))) GO TO 10
C
C     THE DIFFERENCES OF THE Z ARE INCREASING AND EXTRAPOLATION
C     WILL NOT BE PERFORMED.  SAVE ONLY THE LAST TWO ESTIMATES.
C
      Z(1)=Z(J-1)
      Z(2)=Z(J)
      J=2
C
C     I WILL MARK THE BEGINNING OF THE W ARRAY.  W WILL CONTAIN
C     THE SUCCESSIVE COLUMNS OF THE E-SUB-1 TRANSFORMATION.
C
10    I=1
      DO 20 K=1,J
      W(K)=Z(K)
20    CONTINUE
C
C     TRANSFORMATION LOOP.  FIRST, CHECK IF CURRENT TRANSFORMATION
C     IS ACCURATE ENOUGH, BASED ON BEST TWO VALUES OF W.
C
30    E=W(J)-W(J-1)
      IF (E.EQ.0.D0) GO TO 50
      IF (IACC.EQ.0) GO TO 40
      IF (W(J).EQ.0.D0) GO TO 90
      E=E/W(J)
40    IF (DABS(E).GT.ACC) GO TO 60
C
C     THE TEST HAS PASSED.  RETURN THE BEST ESTIMATE
C
50    Z(J)=W(J)
      IER=L
      RETURN
C
```

```
C     THE TEST FAILED.  PERFORM THE E-SUB-1 TRANSFORMATION IF THERE     ERRA0590
C     ARE AT LEAST FOUR ELEMENTS IN W.                                  ERRA0600
C                                                                       ERRA0610
60      I=I+2                                                           ERRA0620
        IF (J.LE.I) GO TO 90                                            ERRA0630
        DO 70 K=I,J                                                     ERRA0640
        M=J+I-K                                                         ERRA0650
        D=W(M)-W(M-1)                                                   ERRA0660
        E=-W(M-1)+W(M-2)+D                                              ERRA0670
        IF (E.EQ.0.D0) GO TO 80                                         ERRA0680
        W(M)=W(M)-D*D/E                                                 ERRA0690
70      CONTINUE                                                        ERRA0700
        GO TO 30                                                        ERRA0710
C                                                                       ERRA0720
C     THE TRANSFORMATION CANNOT BE COMPLETED.  REMOVE THE               ERRA0730
C     SEQUENCE OF THE W ARRAY CAUSING TROUBLE AND TRY AGAIN.            ERRA0740
C                                                                       ERRA0750
80      I=K+1                                                           ERRA0760
        GO TO 30                                                        ERRA0770
C                                                                       ERRA0780
C     THE ERROR CHECK FAILS AND AN E-SUB-1 TRANSFORMATION               ERRA0790
C     EITHER CANNOT BE MADE OR WILL NOT HELP (I.E., ESTIMATE            ERRA0800
C     IS EXACTLY ZERO AND A RELATIVE ERROR TOLERANCE IS SPECIFIED).     ERRA0810
C     CHECK IF THE Z ARRAY IS FULL BEFORE RETURNING.                    ERRA0820
C                                                                       ERRA0830
90      IF (J.NE.20) GO TO 110                                          ERRA0840
        DO 100 I=1,19                                                   ERRA0850
        Z(I)=Z(I+1)                                                     ERRA0860
100     CONTINUE                                                        ERRA0870
        J=19                                                            ERRA0880
110     IER=-L                                                          ERRA0890
        RETURN                                                          ERRA0900
        END                                                             ERRA0910
```

```
      SUBROUTINE EVAL(M1,M2,WEIGHT,FCT,FCT1,ESTIM,IER)
C     PURPOSE
C       THIS SUBROUTINE COMPUTES THE INTEGRAL OF WEIGHT(X)*FCT(X) FROM
C       BASE+2*PI*M1 TO BASE+2*PI*M2 USING A 10 POINT GAUSSIAN INTEGRA-
C       TION SCHEME WITH RESPECT TO WEIGHT(X).  IF THE INTERVAL IS LESS
C       THAN SIX CYCLES LONG, THE INTEGRAL IS COMPUTED USING 'FOGIE'.
C
C     ARGUMENTS
C       M1     - THE LOWER LIMIT OF INTEGRATION (WHEN MULTIPLIED BY 2*PI)
C       M2     - THE UPPER LIMIT OF INTEGRATION (WHEN MULTIPLIED BY 2*PI)
C       WEIGHT - NAME OF A SUBROUTINE TO COMPUTE ABSCISSAE AND WEIGHTS
C                FOR INTEGRATION WITH RESPECT TO A WEIGHTING FUNCTION.
C       FCT    - NAME OF A SUBROUTINE TO COMPUTE F(X).
C       FCT1   - NAME OF A SUBROUTINE TO COMPUTE F(X)*W(X).
C       ESTIM  - REAL*8 OUTPUT.  THE ESTIMATED INTEGRAL.
C       IER    - INTEGER*4 OUTPUT.  ERROR CODE.  IF
C                    IER= L - ESTIM CONTAINS THE ESTIMATE.
C                    IER=-L - NO ESTIMATE - NUMBER OF FUNCTION
C                             EVALUATIONS WERE ABOUT TO EXCEED LIMIT.
C
      COMMON /CMWGT/M/CMSPLT/BASE,LTOTAL,L,ACC,LIM
      EXTERNAL FCT1
      REAL*8 X(10),W(10),ACC,BASE,Y,ESTIM,PI,BOTTOM
      DATA PI/3.141592653589793238462643D0/
      ESTIM=0.D0
      IF (M1.EQ.M2) GO TO 30
      BOTTOM=BASE+2.D0*PI*DFLOAT(M1)
      IF (M2-M1.LT.6) GO TO 50
      IF (L+10.GT.LIM) GO TO 40
      IF (M.EQ.M2-M1) GO TO 10
      M=M2-M1
      CALL WEIGHT(X,W)
10    DO 20 I=1,10
      CALL FCT(X(I)+BOTTOM,Y)
      ESTIM=ESTIM+W(I)*Y
20    CONTINUE
      L=L+10
30    IER=L
      RETURN
40    IER=-L
      RETURN
50    CALL FOGIE(FCT1,5,1,BOTTOM,BASE+2.D0*PI*DFLOAT(M2),ACC,1,LIM-L,
     *           ESTIM,IER)
      L=L+IABS(IER)
      IER=ISIGN(L,IER)
      RETURN
      ENTRY INIT
      M=0
      RETURN
      END
      SUBROUTINE COSWT(X,W)                                           CSWT0010
C     PURPOSE                                                         CSWT0020
C       THIS SUBROUTINE LOOKS UP ABSCISSAE X AND WEIGHTS W FOR 10-POINT CSWT0030
C       GAUSSIAN INTEGRATION WITH WEIGHTING FUNCTION 1+COS(X) FOR     CSWT0040
C       6, 7, 8, 9, 10, 20, 40, 80, 160, AND 320 CYCLES IN THE INTERVAL CSWT0050
C       OF INTEGRATION.                                               CSWT0060
C                                                                     CSWT0070
C     ARGUMENTS                                                       CSWT0080
C       X - REAL*8 OUTPUT VECTOR.  ON OUTPUT, X(I) CONTAINS THE       CSWT0090
C           I-TH ABSCISSA FOR NUMERICAL INTEGRATION. MUST BE          CSWT0100
C           DIMENSIONED 10 IN THE CALLING PROGRAM.                    CSWT0110
C       W - REAL*8 OUTPUT VECTOR.  ON OUTPUT, W(I) CONTAINS THE       CSWT0120
C           I-TH WEIGHT FOR NUMERICAL INTEGRATION.  MUST BE DIMENSIONED CSWT0130
C           10 IN THE CALLING PROGRAM.                                CSWT0140
C                                                                     CSWT0150
      COMMON /CNWGT/M                                                 CSWT0160
      REAL*8 X(10),W(10)                                              CSWT0170
      IF (M.EQ.320) GO TO 90                                          CSWT0180
      IF (M.EQ.160) GO TO 80                                          CSWT0190
      IF (M.EQ.80) GO TO 70                                           CSWT0200
      IF (M.EQ.40) GO TO 60                                           CSWT0210
      IF (M.EQ.20) GO TO 50                                           CSWT0220
      IF (M.EQ.10) GO TO 40                                           CSWT0230
      IF (M.EQ.9) GO TO 30                                            CSWT0240
      IF (M.EQ.8) GO TO 20                                            CSWT0250
      IF (M.EQ.7) GO TO 10                                            CSWT0260
      CALL COS6(X,W)                                                  CSWT0270
      RETURN                                                          CSWT0280
10    CALL COS7(X,W)                                                  CSWT0290
      RETURN                                                          CSWT0300
20    CALL COS8(X,W)                                                  CSWT0310
      RETURN                                                          CSWT0320
30    CALL COS9(X,W)                                                  CSWT0330
      RETURN                                                          CSWT0340
40    CALL COS10(X,W)                                                 CSWT0350
      RETURN                                                          CSWT0360
50    CALL COS20(X,W)                                                 CSWT0370
      RETURN                                                          CSWT0380
60    CALL COS40(X,W)                                                 CSWT0390
      RETURN                                                          CSWT0400
70    CALL COS80(X,W)                                                 CSWT0410
      RETURN                                                          CSWT0420
80    CALL COS160(X,W)                                                CSWT0430
      RETURN                                                          CSWT0440
90    CALL COS320(X,W)                                                CSWT0450
      RETURN                                                          CSWT0460
      END                                                             CSWT0470
      SUBROUTINE COS6(XA,WA)                                          CSWT0480
      REAL*8 XA(1),WA(1),X(10),W(10)                                  CSWT0490
      DATA X/3.7300523025798046461432D1,3.5704261838470024770231D1,   CSWT0500
     *     3.1573517713853618715802D1,2.6206799326922705972720D1,     CSWT0510
     *     2.1219549002903601575554D1,1.6479562840114954980208D1,     CSWT0520
     *     1.1492312516354104445782D1,6.1255941288162771840575D0,     CSWT0530
     *     1.9948500052464246401485D0,3.9858881690779027207050D-1/    CSWT0540
      DATA W/1.9034038965187450422919D0,1.5456150466701007990846D0,   CSWT0550
     *     5.3983226244861548131066D0,5.2134219400476866006096D0,     CSWT0560
     *     4.7887924144353719148184D0,4.7887924129356841929346D0,     CSWT0570
     *     5.2134219415426831338893D0,5.3983226231876183380483D0,     CSWT0580
     *       1.54561504888428052641348D0,1.90034037893659495671680D0/          CS*T0590
      DO 10 I=1,10                                                             CS*T0600
      XA(I)=X(I)                                                               CS*T0610
      WA(I)=W(I)                                                               CS*T0620
10    CONTINUE                                                                 CS*T0630
      RETURN                                                                   CS*T0640
      END                                                                      CS*T0650
      SUBROUTINE COS7(XA,WA)                                                   CS*T0660
      REAL*8 X(10),W(10),XA(1),WA(1)                                           CS*T0670
      DATA X/4.3528064312389415647431D1,4.1503162391104527883898D1,            CS*T0680
     *       3.7518483192584426433314D1,3.1580210757376246277772D1,            CS*T0690
     *       2.5241433749280521859621D1,1.8740863400963132078369D1,            CS*T0700
     *       1.2402086392857029295556D1,6.4638138578352997727251D0,            CS*T0710
     *       2.4781347588341896557200D0,4.5323283805071566676670D-1/           CS*T0720
      DATA W/2.1337922677568312602474D0,1.6036090309380523950196D0,            CS*T0730
     *       5.6110820508366533841042D0,6.1718290213040562681938D0,            CS*T0740
     *       6.4708362041815568365134D0,6.4708362040491966154053D0,            CS*T0750
     *       6.1618290212228062857996D0,5.6110820511004218323552D0,            CS*T0760
     *       1.6036090314228678939494D0,2.1337822447258498956907D0/            CS*T0770
      DO 10 I=1,10                                                             CS*T0780
      XA(I)=X(I)                                                               CS*T0790
      WA(I)=W(I)                                                               CS*T0800
10    CONTINUE                                                                 CS*T0810
      RETURN                                                                   CS*T0820
      END                                                                      CS*T0830
      SUBROUTINE COS8(XA,WA)                                                   CS*T0840
      REAL*8 XA(1),WA(1),X(10),W(10)                                           CS*T0850
      DATA X/4.8734810281789819208598D1,4.6635595436199650976050D1,            CS*T0860
     *       4.2775568039578502066433D1,3.6310245078939370782540D1,            CS*T0870
     *       2.8854131788859095672453D1,2.1311350668442418765380D1,            CS*T0880
     *       1.3855237377564789317042D1,7.4899144179227019613165D0,            CS*T0890
     *       3.6298870210162841321022D0,5.3057217577076833034963D-1/           CS*T0900
      DATA W/2.4307418422520479954758D0,2.4274956747984845033580D0,            CS*T0910
     *       5.6522173435317241124345D0,7.0282947140042360033927D0,            CS*T0920
     *       7.5939916542104801920072D0,7.5939916535124866361702D0,            CS*T0930
     *       7.0282947142928733136882D0,5.6522173439302639133581D0,            CS*T0940
     *       2.4274956743555459658973D0,2.4307415219220896262111D0/            CS*T0950
      DO 10 I=1,10                                                             CS*T0960
      XA(I)=X(I)                                                               CS*T0970
      WA(I)=W(I)                                                               CS*T0980
10    CONTINUE                                                                 CS*T0990
      RETURN                                                                   CS*T1000
      END                                                                      CS*T1010
      SUBROUTINE COS9(XA,WA)                                                   CS*T1020
      REAL*8 XA(1),WA(1),X(10),W(10)                                           CS*T1030
      DATA X/5.5971714312452043316171D1,5.2049722633371282346283D1,            CS*T1040
     *       4.7541883924238593317568D1,4.0664370434213196858764D1,            CS*T1050
     *       3.2553218855989232662180D1,2.4017345868130000578617D1,            CS*T1060
     *       1.5884297329762205208681D1,8.0066838421244228563012D0,            CS*T1070
     *       4.4989451293347220151020D0,5.7695345285493879194938D-1/           CS*T1080
      DATA W/2.5853623056581343853766D0,3.7496043852903983418191D0,            CS*T1090
     *       5.8071788364937026805002D0,7.6812575513860900560130D0,            CS*T1100
     *       8.4509308030394371224960D0,8.4509308047289290654934D0,            CS*T1110
     *       7.6812575484366067471086D0,5.8071768401620563700492D0,            CS*T1120
     *       3.7496043832182807717893D0,2.5853623726047916635950D0/            CS*T1130
      DO 10 I=1,10                                                             CS*T1140
      XA(I)=X(I)                                                               CS*T1150
      WA(I)=W(I)                                                               CS*T1160
```

```
10    CONTINUE                                                          CSWT1170
      RETURN                                                            CSWT1180
      END                                                               CSWT1190
      SUBROUTINE COS10(XA,WA)                                           CSWT1200
      REAL*8 XA(1),WA(1),X(10),W(10)                                    CSWT1210
      DATA X/6.2223115600956301562874D1,5.7840424560877536919179D1,     CSWT1220
     *       5.2448783691958138831524D1,4.4954455947182644592885D1,     CSWT1230
     *       3.6076254731339560066772D1,2.6755598340547294355929D1,     CSWT1240
     *       1.7877597124207334400126D1,1.0382069380689916693328D1,     CSWT1250
     *       4.9914285101016913070995D0,6.0873747111880838200904D-1/    CSWT1260
      DATA W/2.6807766076435632689604D0,4.6772747996232003600731D0,     CSWT1270
     *       6.4545355549860315648986D0,8.3559384004602920469969D0,     CSWT1280
     *     2*9.2474011736074544144362D0,                                CSWT1290
     *       8.3559383985804478422162D0,6.4545355568991315392046D0,     CSWT1300
     *       4.6772747988032949291332D0,2.6807763507972176409384D0/     CSWT1310
      DO 10 I=1,10                                                      CSWT1320
      XA(I)=X(I)                                                        CSWT1330
      WA(I)=W(I)                                                        CSWT1340
10    CONTINUE                                                          CSWT1350
      RETURN                                                            CSWT1360
      END                                                               CSWT1370
      SUBROUTINE COS20(XA,WA)                                           CSWT1380
      REAL*8 XA(1),WA(1),X(10),W(10)                                    CSWT1390
      DATA X/1.2456784992863089200910D2,1.1738762023385587376400D2,     CSWT1400
     *       1.0561972718877768784296D2,9.0119516629531906584513D1,     CSWT1410
     *       7.2204769702539859821400D1,5.3456936441384139470756D1,     CSWT1420
     *       3.5544189512466459746065D1,2.0043978957260222275362D1,     CSWT1430
     *       8.2760859072011829784543D0,1.0958562160063427537438D0/     CSWT1440
      DATA W/3.8509781097021100353660D0,9.5999529878379975896762D0,     CSWT1450
     *       1.3818075792506995208064D1,1.6956821972705333206266D1,     CSWT1460
     *     2*1.8606024210785204507218D1,                                CSWT1470
     *       1.6956821965924448610933D1,1.3818075798482277595516D1,     CSWT1480
     *       9.5999529851388245764070D0,3.8509782861358575980453D0/     CSWT1490
      DO 10 I=1,10                                                      CSWT1500
      XA(I)=X(I)                                                        CSWT1510
      WA(I)=W(I)                                                        CSWT1520
10    CONTINUE                                                          CSWT1530
      RETURN                                                            CSWT1540
      END                                                               CSWT1550
      SUBROUTINE COS40(XA,WA)                                           CSWT1560
      REAL*8 XA(1),WA(1),X(10),W(10)                                    CSWT1570
      DATA X/2.4850908083550483240742D2,2.3484181940512500474938D2,     CSWT1580
     *       2.1140207060615648515522D2,1.8035438092215257910311D2,     CSWT1590
     *       1.4445012189494107346377D2,1.0687729039253812501887D2,     CSWT1600
     *       7.0973631364158202904946D1,3.9925341681996407317001D1,     CSWT1610
     *       1.6485592881721775171400D1,2.8183314516221407153296D0/     CSWT1620
      DATA W/7.8585903048032408421660D0,1.8880423013313807111140D1,     CSWT1630
     *       2.7652340661124235988093D1,3.3980130042517552596513D1,     CSWT1640
     *     2*3.7292222123365465423482D1,                                CSWT1650
     *       3.3980130036872799196270D1,2.7652340664264956730238D1,     CSWT1660
     *       1.8880423013423670235810D1,7.8585900664713951523196D0/     CSWT1670
      DO 10 I=1,10                                                      CSWT1680
      XA(I)=X(I)                                                        CSWT1690
      WA(I)=W(I)                                                        CSWT1700
10    CONTINUE                                                          CSWT1710
      RETURN                                                            CSWT1720
      END                                                               CSWT1730
      SUBROUTINE COS80(XA,WA)                                           CSWT1740
      REAL*8 XA(1),WA(1),X(10),W(10)                                    CSWT1750
      DATA X/4.9636056017714724930556D2,4.6305960607037516040700D2,     CSWT1760
     *      4.2233744133426524398698 6D2,3.6041573984717831788294D2,    CSWT1770
     *      2.8880012611540706757[illegible]D2,[illegible]1698457858477 7214D2, CSWT1780
     *      1.422390[illegible]673220611[illegible]474[illegible],[illegible]0317184218[illegible]11141 23701, CSWT1790
     *      1.[illegible]1651702[illegible]11171[illegible],[illegible]01 [illegible]4007604 74 [illegible]D2/ CSWT1800
      DATA W/1.6420181702119982[illegible]1D1,[illegible]70035753744.69[illegible]1204D1, CSWT1810
     *      5.5142200333341575891 5291,6.777[illegible]5073264[illegible]97157401, CSWT1820
     *     2*7.4385467650198304751 [illegible]1D1,                      CSWT1830
     *      6.7775550779152670663[illegible]D1,[illegible]96996039780[illegible]0574D1, CSWT1840
     *      3.7600375392589497321408D1,1.64[illegible]1705[illegible]7796188 4D1/ CSWT1850
      DO 10 I=1,10                                                      CSWT1860
      XA(I)=X(I)                                                        CSWT1870
      WA(I)=W(I)                                                        CSWT1880
10    CONTINUE                                                          CSWT1890
      RETURN                                                            CSWT1900
      END                                                               CSWT1910
      SUBROUTINE COS160(XA,WA)                                          CSWT1920
      REAL*8 XA(1),WA(1),X(10),W(10)                                    CSWT1930
      DATA X/9.9233000714809759301716D2,9.37[illegible]142471[illegible]337[illegible]25D2, CSWT1940
     *      8.4430285176105099798871D2,7.2659303708404462440740D2,      CSWT1950
     *      5.7751822449804961934206[illegible],4.2778142465200385458957D2, CSWT1960
     *      2.8471661205914006131934[illegible],1.61006797398797814[illegible]352D2, CSWT1970
     *      6.7655224340453102627180D1,1.2579642005695565[illegible]677541/ CSWT1980
      DATA W/3.3336013541060270615964D1,7.5139651368002760074 9D1,      CSWT1990
     *      1.1016743250813762041897D2,1.354035687337826071[illegible]10D2, CSWT2000
     *     2*1.4866081584499454357328 4D2,                              CSWT2010
     *      1.3540356870630416563042D2,1.1016743250181[illegible]795266 52D2, CSWT2020
     *      7.5139651370886505172382D1,3.3336[illegible]123661[illegible]9004[illegible]2274D1/ CSWT2030
      DO 10 I=1,10                                                      CSWT2040
      XA(I)=X(I)                                                        CSWT2050
      WA(I)=W(I)                                                        CSWT2060
10    CONTINUE                                                          CSWT2070
      RETURN                                                            CSWT2080
      END                                                               CSWT2090
      SUBROUTINE COS320(XA,WA)                                          CSWT2100
      REAL*8 XA(1),WA(1),X(10),W(10)                                    CSWT2110
      DATA X/1.9844560348127176894195D3,1.8750534714023[illegible]29769D3, CSWT2120
     *      1.6883979513143567601219D3,1.4410522271170853[illegible]074D3, CSWT2130
     *      1.1549803215014830948348D3,8.5562897679817274305632D2,      CSWT2140
     *      5.6956707117288624431239D2,3.22271344914983605 3274D2,      CSWT2150
     *      1.3556582688475252140847D2,2.6165263488552679852979D1/      CSWT2160
      DATA W/6.6935672076864142279964D1,1.5025314694880120290309D2,     CSWT2170
     *      2.2027120144182825185.04D2,2.7072472150748117104942D2,      CSWT2180
     *     2*2.8712481718447318900082D2,                                CSWT2190
     *      2.7072472146747690534640D2,2.2027128146842574045752D2,      CSWT2200
     *      1.5025314694889425117049D2,6.6935673311887719297382D1/      CSWT2210
      DO 10 I=1,10                                                      CSWT2220
      XA(I)=X(I)                                                        CSWT2230
      WA(I)=W(I)                                                        CSWT2240
10    CONTINUE                                                          CSWT2250
      RETURN                                                            CSWT2260
      END                                                               CSWT2270
```

THIS PAGE IS BEST QUALITY PRACTICABLE
FROM COPY FURNISHED TO DDC

```
      SUBROUTINE SINWT(X,W)
C   PURPOSE
C     THIS SUBROUTINE LOOKS UP ABSCISSAE X AND WEIGHTS W FOR 10-POINT
C     GAUSSIAN INTEGRATION WITH WEIGHTING FUNCTION 1+SIN(X) FOR
C     6, 7, 8, 9, 10, 20, 40, 80, 160, AND 320 CYCLES IN THE INTERVAL
C     OF INTEGRATION.
C
C   ARGUMENTS
C     X - REAL*8 OUTPUT VECTOR.  ON OUTPUT, X(I) CONTAINS THE
C         I-TH ABSCISSA FOR NUMERICAL INTEGRATION. MUST BE
C         DIMENSIONED 10 IN THE CALLING PROGRAM.
C     W - REAL*8 OUTPUT VECTOR.  ON OUTPUT, W(I) CONTAINS THE
C         I-TH WEIGHT FOR NUMERICAL INTEGRATION.  MUST BE DIMENSIONED
C         10 IN THE CALLING PROGRAM.
C
C
      COMMON /CMWGT/M
      REAL*8 X(10),W(10)
      IF (M.EQ.320) GO TO 90
      IF (M.EQ.160) GO TO 80
      IF (M.EQ.80) GO TO 70
      IF (M.EQ.40) GO TO 60
      IF (M.EQ.20) GO TO 50
      IF (M.EQ.10) GO TO 40
      IF (M.EQ.9) GO TO 30
      IF (M.EQ.8) GO TO 20
      IF (M.EQ.7) GO TO 10
      CALL SIN6(X,W)
      RETURN
10    CALL SIN7(X,W)
      RETURN
20    CALL SIN8(X,W)
      RETURN
30    CALL SIN9(X,W)
      RETURN
40    CALL SIN10(X,W)
      RETURN
50    CALL SIN20(X,W)
      RETURN
60    CALL SIN40(X,W)
      RETURN
70    CALL SIN80(X,W)
      RETURN
80    CALL SIN160(X,W)
      RETURN
90    CALL SIN320(X,W)
      RETURN
      END
      SUBROUTINE SIN6(XA,WA)
      REAL*8 XA(1),WA(1),X(10),W(10)
      DATA X/3.7386307692880386355228D1,3.4300841050987418867635D1,
     *       3.1864591866425745791958D1,2.6723164277940480104689D1,
     *       2.0835171766438769225260D1,1.4991107848691171966493D1,
     *       1.0076456296502385612257D1,6.6347737793769390179932D0,
     *       2.1345819558929777226552D0,5.1203806918112540580523D-1/
      DATA W/5.4423087531894081171154D-1,2.6665291031280396639847D0,
     *       3.8996322503223345955186D0,5.7877211754834574768807D0,
     *       5.9292919630340323896817D0,5.6464196050403216026917D0,
     *      3.8126600667511662014766D0,3.9830904024563448842605D0,          SNWT0590
     *      3.6453790822613281085576D0,1.7741573263304367864521D0/          SNWT0600
      DO 10 I=1,10                                                          SNWT0610
      XA(I)=X(I)                                                            SNWT0620
      WA(I)=W(I)                                                            SNWT0630
10    CONTINUE                                                              SNWT0640
      RETURN                                                                SNWT0650
      END                                                                   SNWT0660
      SUBROUTINE SIN7(XA,WA)                                                SNWT0670
      REAL*8 XA(1),WA(1),X(10),W(10)                                        SNWT0680
      DATA X/4.3615615243916813881242D1,4.0130426563793609062714D1,         SNWT0690
     *      3.6882424279362880473521D1,3.1822526782213207496852D1,          SNWT0700
     *      2.5641982874851841245345D1,1.9214118463509465646212D1,          SNWT0710
     *      1.3124569325094626037753D1,7.4417157104936451084143D0,          SNWT0720
     *      2.3577724238342567808502D0,5.8442127962048757983116D-1/         SNWT0730
      DATA W/5.8233419281745102982340D-1,4.0534631997777487157590D0,        SNWT0740
     *      3.5187989607796137400587D0,5.9564693910321925804638D0,          SNWT0750
     *      6.4566745190409642625739D0,6.3154162451130583513838D0,          SNWT0760
     *      5.8516957786700427313944D0,5.5555720736066269878123D0,          SNWT0770
     *      3.6097196939555663597875D0,2.0711520669698407957424D0/          SNWT0780
      DO 10 I=1,10                                                          SNWT0790
      XA(I)=X(I)                                                            SNWT0800
      WA(I)=W(I)                                                            SNWT0810
10    CONTINUE                                                              SNWT0820
      RETURN                                                                SNWT0830
      END                                                                   SNWT0840
      SUBROUTINE SIN8(XA,WA)                                                SNWT0850
      REAL*8 XA(1),WA(1),X(10),W(10)                                        SNWT0860
      DATA X/4.9833314124884104501234D1,4.6109690669691250519691D1,         SNWT0870
     *      4.1334931454417528584589D1,3.5715485804653418711041D1,          SNWT0880
     *      2.8803430374492685165145D1,2.1443208762570648701274D1,          SNWT0890
     *      1.4334053190126041954500D1,7.8945321014878477869306D0,          SNWT0900
     *      2.6241962561975662460156D0,6.6314099103032231318332D-1/         SNWT0910
      DATA W/6.4811911161963082437184D-1,4.9281743745325980057354D0,        SNWT0920
     *      4.8816179283342269166625D0,6.3835345696415724052184D0,          SNWT0930
     *      7.2716039845500804500464D0,7.3325362895640068676857D0,          SNWT0940
     *      6.8127457441010430937845D0,6.0519048290612034675987D0,          SNWT0950
     *      3.5282546606554703753374D0,2.4069909707320020828628D0/          SNWT0960
      DO 10 I=1,10                                                          SNWT0970
      XA(I)=X(I)                                                            SNWT0980
      WA(I)=W(I)                                                            SNWT0990
10.   CONTINUE                                                              SNWT1000
      RETURN                                                                SNWT1010
      END                                                                   SNWT1020
      SUBROUTINE SIN9(XA,WA)                                                SNWT1030
      REAL*8 XA(1),WA(1),X(10),W(10)                                        SNWT1040
      DATA X/5.6051760098161565792807D1,5.2209139847414316903851D1,         SNWT1050
     *      4.6585773717309384664986D1,3.9877543805063002224642D1,          SNWT1060
     *      3.2012775706700892887061D1,2.3697896613964179124423D1,          SNWT1070
     *      1.5677088706848231915287D1,8.5430058091850397828441D0,          SNWT1080
     *      3.0988312240454518587943D0,7.6898054053188512669542D-1/         SNWT1090
      DATA W/7.0191084613106270157754D-1,5.3322739173058335753698D0,        SNWT1100
     *      6.0461343088249275580211D0,7.3832175792102893375227D0,          SNWT1110
     *      8.2194115527488834502354D0,8.2842576758365047267068D0,          SNWT1120
     *      7.6560162595357626287800D0,6.5444142708977522080825D0,          SNWT1130
     *      3.4981301115940463465218D0,2.8829012827128600765836D0/          SNWT1140
      DO 10 I=1,10                                                          SNWT1150
      XA(I)=X(I)                                                            SNWT1160
      WA(I)=W(I)                                                        SNWT1170
   10 CONTINUE                                                          SNWT1180
      RETURN                                                            SNWT1190
      END                                                               SNWT1200
      SUBROUTINE SIN10(XA,WA)                                           SNWT1210
      REAL*8 XA(1),WA(1),X(10),W(10)                                    SNWT1220
      DATA X/6.2251045261232956562480D1,5.8306319930145448893200D1,     SNWT1230
     *     5.2073405449835263425484D1,4.4413019828682479328563D1,       SNWT1240
     *     3.5556391593225036729109D1,2.6250204824769753031433D1,       SNWT1250
     *     1.7309267803954298869940D1,9.4717013503330722556939D0,       SNWT1260
     *     3.7865818791779763952121D0,8.7521046726596177234292D-1/      SNWT1270
      DATA W/7.7268076070697193351106D-1,5.6632258042307027295044D0,    SNWT1280
     *     6.9040853344198017082448D0,8.3639907441764233332700D0,       SNWT1290
     *     9.2184344844166148810044D0,9.2561454163305461608482D0,       SNWT1300
     *     8.5022382934843015347575D0,7.0544987206836163589760D0,       SNWT1310
     *     3.7269621118746238484754D0,3.3695812999761663790574D0/       SNWT1320
      DO 10 I=1,10                                                      SNWT1330
      XA(I)=X(I)                                                        SNWT1340
      WA(I)=W(I)                                                        SNWT1350
   10 CONTINUE                                                          SNWT1360
      RETURN                                                            SNWT1370
      END                                                               SNWT1380
      SUBROUTINE SIN20(XA,WA)                                           SNWT1390
      REAL*8 XA(1),WA(1),X(10),W(10)                                    SNWT1400
      DATA X/1.2338167390777451258964D2,1.1716429469531214735412D2,     SNWT1410
     *     1.0551917122195482079405D2,9.0076026472154495650102D1,       SNWT1420
     *     7.2223942947759507404726D1,5.3546598886794527061284D1,       SNWT1430
     *     3.5704737827421183737897D1,2.0290260286474200768225D1,       SNWT1440
     *     8.6742392404773605019130D0,1.4073606531852308595113D0/       SNWT1450
      DATA W/3.2858176515055225748958D0,9.2992491275906254137065D0,     SNWT1460
     *     1.3758259189463008171278D1,1.6897867103528076456403D1,       SNWT1470
     *     1.8539980583571882905302D1,1.8535446129093148392597D1,       SNWT1480
     *     1.6880667505695339344751D1,1.3715434809200916197816D1,       SNWT1490
     *     9.3783293958914219414854D0,5.3746551036564661365220D0/       SNWT1500
      DO 10 I=1,10                                                      SNWT1510
      XA(I)=X(I)                                                        SNWT1520
      WA(I)=W(I)                                                        SNWT1530
   10 CONTINUE                                                          SNWT1540
      RETURN                                                            SNWT1550
      END                                                               SNWT1560
      SUBROUTINE SIN40(XA,WA)                                           SNWT1570
      REAL*8 XA(1),WA(1),X(10),W(10)                                    SNWT1580
      DATA X/2.4696796662280731027295D2,2.3344479391196871276283D2,     SNWT1590
     *     2.1020726423274027183652D2,1.7937204586524208593801D2,       SNWT1600
     *     1.4370025637217043978922D2,1.0636629193833022810622D2,       SNWT1610
     *     7.0688225401885972587479D1,3.9833844522231149909430D1,       SNWT1620
     *     1.6529055698539099950040D1,2.6565132959117647715175D0/       SNWT1630
      DATA W/8.3707313203736708648774D0,1.8662383932853529037794D1,     SNWT1640
     *     2.7450693754708714895127D1,3.3756283407357596418513D1,       SNWT1650
     *     3.7053597366904011243080D1,3.7056353884339116898925D1,       SNWT1660
     *     3.3766078357437136698714D1,2.7482498279003257967147D1,       SNWT1670
     *     1.8793938265813315146594D1,8.9339537867492147934291D0/       SNWT1680
      DO 10 I=1,10                                                      SNWT1690
      XA(I)=X(I)                                                        SNWT1700
      WA(I)=W(I)                                                        SNWT1710
   10 CONTINUE                                                          SNWT1720
      RETURN                                                            SNWT1730
      END                                                               SNWT1740
      SUBROUTINE SIN80(XA,WA)                                             SNWT1750
      REAL*8 XA(1),WA(1),X(10),W(10)                                      SNWT1760
      DATA X/4.8500259632396875986160D2,4.6765986852625979963705D2,       SNWT1770
     *      4.2104060141771333282127D2,3.5826006541836248050195D2,        SNWT1780
     *      2.8780808568628128796263D2,2.1303344044292882131231D2,        SNWT1790
     *      1.4157958326991774811177D2,7.9783475982000686030915D1,        SNWT1800
     *      3.3156146505452230854871D1,5.7276955655123531485629D0/        SNWT1810
      DATA W/1.6855408605165703736988D1,3.7525047820554478472825D1,       SNWT1820
     *      5.5017162662723643430404D1,6.7620686047349228878556D1,        SNWT1830
     *      7.4215621123390187379566D1,7.4216451618718057730945D1,        SNWT1840
     *      6.7623861805255503782928D1,5.5026221483288409134054D1,        SNWT1850
     *      3.7558661240419813758009D1,1.6995702274401745306525D1/        SNWT1860
      DO 10 I=1,10                                                        SNWT1870
      XA(I)=X(I)                                                          SNWT1880
      WA(I)=W(I)                                                          SNWT1890
10    CONTINUE                                                            SNWT1900
      RETURN                                                              SNWT1910
      END                                                                 SNWT1920
      SUBROUTINE SIN160(XA,WA)                                            SNWT1930
      REAL*8 XA(1),WA(1),X(10),W(10)                                      SNWT1940
      DATA X/9.9113491342491528257597D2,9.3641703891555903283006D2,       SNWT1950
     *      8.4311544689642323646684D2,7.1948087810003198683262D2,        SNWT1960
     *      5.7649487990145728554570D2,4.2686125809032324696091D2,        SNWT1970
     *      2.8387478794637536338996D2,1.6023882365368494973268D2,        SNWT1980
     *      6.6932750061035008570798D1,1.2193686001252605377587D1/        SNWT1990
      DATA W/3.3583914127021666649216D1,7.5109751855573192798477D1,       SNWT2000
     *      1.1010226573423026747439D2,1.3531897275127145494190D2,        SNWT2010
     *      1.4851564164674830540871D2,1.4851585062314187012043D2,        SNWT2020
     *      1.3532077028188800251755D2,1.1010452807865155864059D2,        SNWT2030
     *      7.5118042717647241043443D1,3.3618908240611396844433D1/        SNWT2040
      DO 10 I=1,10                                                        SNWT2050
      XA(I)=X(I)                                                          SNWT2060
      WA(I)=W(I)                                                          SNWT2070
10    CONTINUE                                                            SNWT2080
      RETURN                                                              SNWT2090
      END                                                                 SNWT2100
      SUBROUTINE SIN320(XA,WA)                                            SNWT2110
      REAL*8 XA(1),WA(1),X(10),W(10)                                      SNWT2120
      DATA X/1.9833552710669182488346D3,1.8739275134964322546693D3,       SNWT2130
     *      1.6872965048289770483284D3,1.4399889249781799662742D3,        SNWT2140
     *      1.1539723172896959306147D3,8.5465858651525303457674D2,        SNWT2150
     *      5.6864186098808568647894D2,3.2133393287768791424241D2,        SNWT2160
     *      1.3470180779590307906801D2,2.5268770145658663750510D1/        SNWT2170
      DATA W/6.7065863320014378106617D1,1.5023972880682882387257D2,       SNWT2180
     *      2.2023850807280241203432D2,2.7068216445425884190606D2,        SNWT2190
     *      2.9707757224165065459623D2,2.9707762440368697484614D2,        SNWT2200
     *      2.7068236361316044884730D2,2.2023807359543380305824D2,        SNWT2210
     *      1.5024179201740014386769D2,6.7074604612436680231078D1/        SNWT2220
      DO 10 I=1,10                                                        SNWT2230
      XA(I)=X(I)                                                          SNWT2240
      WA(I)=W(I)                                                          SNWT2250
10    CONTINUE                                                            SNWT2260
      RETURN                                                              SNWT2270
      END                                                                 SNWT2280
      SUBROUTINE FCOSXA(X,F)
C    PURPOSE
C      THIS SUBROUTINE COMPUTES F(X)*(1+COS(X-A)).
C
      COMMON /CMSPLT/A/CMIF/FOFX
      REAL*8 X,F,A
      CALL FLINK(FOFX,X,F)
      F=F+F*DCOS(X-A)
      RETURN
      END
      SUBROUTINE FSINXA(X,F)
C    PURPOSE
C      THIS SUBROUTINE COMPUTES F(X)*(1+SIN(X-A)).
C
      COMMON /CMSPLT/A/CMIF/FOFX
      REAL*8 X,F,A
      CALL FLINK(FOFX,X,F)
      F=F+F*DSIN(X-A)
      RETURN
      END
      SUBROUTINE FCOSX(X,F)
C    PURPOSE
C      THIS SUBROUTINE COMPUTES F(X)*COS(X)
C
      COMMON /CMIF/FOFX
      REAL*8 X,F
      CALL FLINK(FOFX,X,F)
      F=F*DCOS(X)
      RETURN
      END
      SUBROUTINE FSINX(X,F)
C    PURPOSE
C      THIS SUBROUTINE COMPUTES F(X)*SIN(X)
C
      COMMON /CMIF/FOFX
      REAL*8 X,F
      CALL FLINK(FOFX,X,F)
      F=F*DSIN(X)
      RETURN
      END
      SUBROUTINE FLINK(FOFX,X,F)
C    PURPOSE
C      THIS SUBROUTINE PROVIDES LINKAGE TO THE USER SUBROUTINE 'FOFX'
C
      REAL*8 X,F
      CALL FOFX(X,F)
      RETURN
      END
```

COMPUTER–AIDED SOLUTION OF THE BACTERIAL SURVIVAL EQUATIONS IN MICROBIOLOGY

Chia Ping Wang and Ari Brynjolfsson
US Army Natick Research and Development Command
Natick, Massachusetts 01760

ABSTRACT. It is shown that the survival fraction N/N_o of a microorganism during thermal sterilization may be represented quite generally by the equation of the form

$$\ell n \frac{N}{N_o} = -C \int_o^t F(T(t),\ t)\ dt$$

where $C \cdot F(T(t),\ t)$ is the kill rate for each microorganism, $F(T(t),\ t)$ is a suitably chosen function of the sterilization temperature $T(t)$ and the time t, and C is a constant.

The equation is solved rather simply with the aid of a computer. The technique is exemplified by solving the equation completely for the case of convective heating in an infinitely long cylindrical specimen and cylinders of finite length.

1. DERIVATION OF THE BACTERIAL SURVIVAL EQUATIONS.

For a large class of microorganisms, the survival fraction N/N_o during thermal sterilization at temperature T follows closely the equation [1–3]

$$\ell n \frac{N}{N_o} = -C \cdot [\exp\ (-E_a/RT)] \cdot t \qquad (1)$$

where
N = the number of bacteria survived at time t
N_o = initial value of N
E_a = the Arrhenius activation energy in kcal-mol^{-1} for inactivating a unit or a molecule that is essential for the survival of a bacterium
R = the universal gas constant in $\text{J} \cdot \text{mol}^{-1}$
T = sterilization temperature in kelvin
t = sterilization time in s
C = a constant

It may be more expedient to express the empirical law (1) in terms of the kill rate

$$\frac{dN}{dt} = -C \cdot [\exp\ (-E_a/RT)] \cdot N \qquad (2)$$

If we take Eq (2) as the empirical law, then for T varying with time t, we have the following time integral

$$\ell n \frac{N}{N_o} = -C \cdot \int_o^t \exp(-E_a/RT) \cdot dt \tag{3}$$

There are other kinds of microorganisms, for which the semilogarithmic plot of the survival fraction is not a straight line but has a "shoulder", i.e., the fractional kill rate (dN/dt)/N is not a constant even though the temperature is held constant. In such a case, the exponential in Eqs (1) and (2) is to be replaced by a general function F(T(t),t), and therefore we have the following generalized equation for the survival fraction

$$\ell n(N/N_o) = -C \cdot \int_o^t F(T(t),\ t)dt \tag{3a}$$

We have shown above that Eq (3) or (3a) for the survival fraction follows directly from the differential kill rate Eq (2). One may argue that the kill rate is a derived quantity, and thus Eq (2) may not be a good starting point to establish Eq (3) or (3a). However, one could show that Eq (3a) follows from the generalized empirical equation

$$\ell n(N/N_o) = -C \cdot F(T(t),\ t) \cdot t \tag{1a}$$

from the mean value theorem [3]. This is given in the Appendix.

From Eq (3a), N/N_o can be determined if F(T(t), t) as a function of T and t is found, and the temperature T(t) of the specimen as a function of the time t is known.

In Section 2, we show how the function T(t) at any point in an infinitely long cylindrical sample is derived from the equations of heat transfer, and in Section 3 we solve the corresponding Eq (3) with the aid of a computer. As will be seen, the complicated situation of conductivity discontinuity at the surface of the sample due to the casing and the adhered surface gas film, and of the radiation exchange at the surface, can all be taken into account by a combined single statement of the boundary condition.

In Section 4, we solve the heat transfer problem and Eq (3) for cylindrical samples of finite length, with the same physical conditions at the boundary as in the case of an infinitely long sample cylinder.

2. TEMPERATURE DISTRIBUTION IN A LONG CYLINDRICAL SPECIMEN. The heat conduction equation for an infinitely long cylinder of radius a is

$$\frac{\partial T}{\partial t} = \kappa \left[\frac{\partial^2 T}{\partial r^2} + \frac{1}{r}\frac{\partial T}{\partial r}\right], \text{ for } 0<r<a \tag{4}$$

where κ is the thermal diffusivity

$$\kappa = \frac{\lambda}{\rho \cdot c} = \frac{\text{Thermal conductivity}}{\text{density} \cdot \text{specific heat}} \tag{4a}$$

Initially the entire cylindrical sample is at temperature T_i. Then suddenly it is exposed to the temperature T_o and the outside temperature is maintained at T_o during the process. The temperature T at any point in the sample will change with time. To simplify the equations, we use a new variable θ where

$$\theta = T - T_o \tag{5}$$

Eq (4) then becomes

$$\frac{\partial \theta}{\partial t} = \kappa \left[\frac{\partial^2 \theta}{\partial r^2} + \frac{1}{r}\frac{\partial \theta}{\partial r}\right] \tag{6}$$

The initial vlaue of θ at any point of the sample is:

$$\theta_i = T_i - T_o \text{ for } t < 0 \tag{7}$$

At the boundary r = a, there will be in practice an interface (layer) or transition layer in which the temperature is changing from T_o to the actual temperature at the surface of the sample. At the surface, the boundary condition of heat flowing into the sample being equal to the heat flowing out from the layer across the surface of the sample, lead to

$$-\frac{\partial \theta}{\partial r} = \frac{h}{\lambda}\,\theta, \; r = a \tag{8}$$

where $h = \lambda_{gas}/\delta_{gas}$, the conductivity of the gas (air) divided by the thickness of the gas (air) film, is the coefficient of heat transfer due to gas film only.

Due to radiation at the surface, h actually would be equal to $\lambda_{gas}/\delta_{gas}$ plus some constant (due to radiation). Also, because of the casing of the sample, we have actually two transition layers instead of one at the surface, the solid casing material and the gas film adhered onto it. We combine these three factors into a single effective coefficient in front of θ in Eq (8). Henceforth, h/λ of Eq (8) will stand for this effective coefficient.

Eq (6) with boundary condition of the form of Eq (8), i.e. $\partial\theta/\partial r$ being proportional to θ, has been solved [3,4] with the usual technique of the separation of variables. Here we use Carslow and Jaeger's notations [4] when applicable. The solution takes the form

$$\theta = \sum_{n=1}^{\infty} A_n J_o(\alpha_n r) \cdot e^{-\kappa \alpha_n^2 t} \tag{9}$$

where $a\alpha_n = x_n$ are roots of

$$x_n \cdot J_1(x_n) = \frac{h}{\lambda} a \cdot J_o(x_n) \tag{10}$$

and

$$A_n = \frac{2\theta_i}{x_n} \cdot \frac{J_1(x_n)}{J_o^2(x_n) + J_1^2(x_n)} \tag{11}$$

In Eqs (9), (10), and (11), J_o and J_1 are Bessel functions of orders zero and one respectively.

Hence, from Eqs (5) and (7),

$$T - T_o = 2(T_i - T_o) \cdot \sum_{n=1}^{\infty} \frac{J_o(\alpha_n r) \cdot e^{-\kappa \alpha_n^2 t}}{x_n[(x_n/\nu)^2 + 1]\ J_1(x_n)} \tag{12}$$

where

$$x_n = a\alpha_n \tag{13}$$

$$\nu = \frac{h}{\lambda} a = \text{effective conduction Nusselt number or Biot number} \tag{14}$$

Had we considered only the gas film at the surface, h in Eqs (14) and (8) would be $\lambda_{gas}/\delta_{gas}$, and ν of Eq (14) would be the "conduction Nusselt number" [5], or the "Biot number", which is to be distinguished from the "convection Nusselt number", the latter being equal $(h/\lambda_{gas})\cdot a$. In our case, (h/λ) is the combined effective coefficient of three factors, the gas film, the solid casing, and the surface radiation. But, it would be meaningful to express the combined effect in terms of an equivalent resistive film at the surface. Thus, the combined constants $(h/\lambda)\cdot a$ of Eq (14) is the equivalent or effective conduction Nusselt number or Biot number.

As a specific example, we carried out a computer search for the ranges of the two parameters κ and ν in Eq (12) for the beef rolls processed in this laboratory, and found that for these, the thermal diffusivity κ lies somewhere around 1.25×10^{-3} to 1.35×10^{-3} cm^{-2} s^{-1}, and the effective Nusselt number ν, around 6 to 8.

Fig. 1 is a computer plot of Eq (12) as a function of time for r = 0, 0.1 a, 0.2 a ... 1.0 a, with $\nu = 6$, $\kappa = 1.35 \times 10^{-3}$ cm^2 s^{-1}, and a = 5 cm. The heating and the initial temperatures are 121.1°C and 21°C respectively.

Fig. 2 is a computer plot of the temperature distribution within the roll for t = 5, 10, 15 ... 240 minutes, i.e., the plot of Eq (12) as a function of r for t = 5, 10, 15 ... 240 minutes, again with $\nu = 6$, $\kappa = 1.35 \times 10^{-3}$ cm^2 s^{-1}, a = 5 cm, and the same heating and initial temperatures as those in Fig. 1.

3. CALCULATIONS OF BACTERIAL SURVIVAL EQUATION.

We now put our function T(t) of Eq (12) into the "bacterial survival integral" of Eqs (3) or (3a) to calculate the survival fraction N/N_0 of the bacterium. In this paper, we carried out the calculations for *Cl. botulinum* spores for which we assume the reaction rate constant to be given by the exponential function $\exp(-E_a/RT)$ rather than the general function F(T(t), t) which may represent any empirical function or other theoretical function such as Eyring's.

The calculations for other microorganisms will be exactly the same.

The integrand of Eq (3) after substituting T from Eq (12) looks quite complicated. We have written computer programs to carry out the computation. But, before we could actually carry out the computation, the two constants C and E_a in Eqs (3) and (1) have to be determined.

3.1 DETERMINATION OF E_a FROM THE z VALUE.

We show in some details below the relationship between the z-value, the temperature T, and the Arrhenius activation energy E_a.

The reaction rate equation, Eq (2),

$$\frac{dN}{dt} = -kN = -C \cdot [\exp(-E_a/RT)] \cdot N \qquad (2)$$

means that the temperature dependence of k is through T in the exponential function $\exp(-E_a/RT)$ only where E_a is a constant independent of T. Otherwise, k will have to be expressed by the general function F(T(t), t). Thus, at temperatures T_1 and T_2, we have

$$(\frac{dN}{dt})_1 = -C \cdot [\exp(-E_a/RT_1)] \cdot N \qquad (15)$$

$$(\frac{dN}{dt})_2 = -C \cdot [\exp(-E_a/RT_2)] \cdot N \qquad (16)$$

From Eqs (15) and (16),

$$\frac{(dN/dt)_1}{(dN/dt)_2} = \exp[-\frac{E_a}{R} \frac{T_2 - T_1}{T_1 T_2}] \qquad (17)$$

Now if we choose the temperature T_2 such that the rate of inactivation is changed by a factor of 10 from that at T_1, then

$$\frac{(dN/dt)_1}{(dN/dt)_2} = 10 \qquad (18)$$

whence

$$E_a = \frac{(\ell n\ 10)\cdot R\ T_1 T_2}{T_1 - T_2} = \frac{(\ell n\ 10)\cdot R\ T_1 T_2}{z} \qquad (19)$$

where, by definition, $T_1 - T_2$, the temperature change as specified, is the z-value.

In case of *Cl. botulinum* spores, representative values for z are in the range z = 10.4 ± 0.8°C at 121.1°C [6]. From Eq (19), we have then for *Cl. botulinum* spores,

$$E_a = \frac{(\ell n\ 10) \cdot 394.26 \cdot 384.26 \cdot 1.987 \cdot 10^{-3}}{10.4}$$

$$= 66.6 \pm 5.4 \text{ kcal. mol}^{-1} \qquad (20)$$

3.2 DETERMINATION OF THE CONSTANT C. For *Cl. botulinum* spores, we will use for 12D (i.e. reduction of the spore number to 10^{-12} of the initial number) the conservative F_O–value of 3.5 min for non-acid and non-cured meats [6], which means that heating at 121.1°C for 17.5 sec. reduces the number N by one order of magnitude. Thus, from Eq (1), for *Cl. botulinum* spores,

$$[C\cdot\exp(-E_a/RT)] \cdot 17.5 = \ell n\ 10$$

giving

$$C = \frac{2.3026}{17.5} \times e^{85.076}$$

$$= 1.17 \cdot 10^{36}\ s^{-1}$$

3.3 SOLUTION OF EQUATION (3). The calculation of the constant C is indeed very simple as given in the above section. We note, however, that C is a very large number, which, in turn, means that the "Bacteria Survival Integral" in Eq (3) will give very small numbers. Certain procedures have been taken to overcome the computer underflow when computing this integral.

Calculations are made rather simply, with the aid of a computer, of Eq (3) as a function of t, with T(t) of Eq (12) inserted in the equation, for $r = 0, 0.1a, 0.2a, 0.3a \ldots 1.0a$. Computations could be performed for any suitably – chosen time intervals. We have used two minute intervals for our present calculations, so that the inverse problem of finding the time t for any given value of the survival fraction N/N_0 can easily be interpolated from the computer output. Fig. 3 is the plot of these "integral" survival curves for *Cl. botulinum* spores with $a = 5$ cm, $\nu = 6$ and $\kappa = 1.35 \times 10^{-3}$ cm^2 s^{-1}. The heating and the initial temperatures, T_0 and T_i, are 121.1°C and 21°C respectively.

It is seen from the survival curve for $r = 0$ that in order to reduce the survival fraction of *Cl. botulinum* spores to 10^{-12} of the initial spore number, 3.5 hours heating is needed for a long beef roll of radius = 5 cm.

4. SOLUTION OF EQUATION (3) FOR FINITE CYLINDRICAL SAMPLES. For finite cylinders with the same physical conditions as in Section 2, the heat conduction equation is

$$\frac{\partial\theta}{\partial t} = \kappa\nabla^2\theta \qquad \text{for } r < a,\ |z| < \ell \tag{22}$$

in cylindrical coordinates, and the boundary and initial conditions are

$$\frac{\partial\theta}{\partial r} + \frac{h}{\lambda}\theta = 0 \qquad \text{at } r = a \tag{23}$$

$$\frac{\partial\theta}{\partial z} + \frac{h}{\lambda}\theta = 0 \qquad \text{at } z = \ell \tag{24}$$

$$-\frac{\partial\theta}{\partial z} + \frac{h}{\lambda}\theta = 0 \qquad \text{at } z = -\ell \tag{25}$$

and $\theta = \theta_i = T_i - T_0$ for $t < 0$ (26)

where ℓ is the half length of the cylinder and all the other symbols have the same meaning as in the previous sections.

The solution for our case, takes the following form

$$\theta = \sum_{\lambda_j} \sum_{\alpha_n} A_{\lambda_j, \alpha_n} \cdot \cos(\lambda_j z) \cdot J_o(\alpha_n r) \cdot e^{-\kappa(\lambda_j^2 + \alpha_n^2)t} \tag{27}$$

where $a\alpha_n$, as before, are roots of Eq (10) and $\lambda_j \ell$ are roots of

$$\frac{h\ell}{\lambda} \cos\lambda_j \ell - \lambda_j \ell \sin\lambda_j \ell = 0 \tag{28}$$

We evaluate the expansion coefficients A_{λ_j, α_n} from the boundary conditions Eqs (23) – (26). The solution for temperature T is then as follows:

$$T - T_o = 2(T_i - T_o) \cdot \sum_{j=1}^{\infty} \sum_{n=1}^{\infty} \frac{1}{x_n[(\frac{x_n}{\nu})^2 + 1] \cdot J_1(x_n)} \cdot J_o(\alpha_n r) \cdot \frac{4 \sin \lambda_j \ell}{\sin 2 \lambda_j \ell + 2 \lambda_j \ell} \cdot \cos\lambda_j z \cdot e^{-\kappa(\lambda_j^2 + \alpha_n^2)t} \tag{29}$$

The function T in Eq (29) is to be inserted in Eq (3) or Eq (3a), and the integration carried out. Though the integrand now becomes quite complex, the computer handles it just as easily as in the infinitely long cylinder case. For a cylinder of length $\ell = a$, the 12D sterilization time is found to reduce to about 80% of the value for the infinitely long roll.

5. CONCULSION. We have derived the following equation for the survival fraction N/N_o of a microorganism during thermal sterilization:

$$\ln \frac{N}{N_o} = -C \int_o^t F(T(t), t)\, dt \tag{3a}$$

where F(T(t), t) may be an exponential or a more general function of temperature T(t) and time t.

The temperature time function T(t), and hence the equation (3a), in general,are quite complex, but with the aid of a computer, the equation can easily be solved. The technique is exemplified by solving the equation completely for the case of convective heating of an infinitely long cylindrical specimen and cylinders of finite length.

APPENDIX

PROOF OF EQUATION (3a) FROM EQUATION (1a)

If at constant temperature T, the logarithmic survival fraction is given by

$$\ell n\ (N/N_O) = -C \cdot F(T,t) \cdot t \qquad (1a)$$

then for a small time interval Δ t at time t_1, there will be a mean effective temperature T_1 and a survival fraction N_1/N_O, such that

$$\ell n\ (N_1/N_O) = -C \cdot F(T_1,\ t_1) \cdot \Delta t$$

Similarly, for Δt at (t_2,T_2) following the first time interval Δt at t_1, and at (t_3,T_3) and so on,

$$\ell n\ (N_2/N_1) = -C \cdot F(T_2,\ t_2) \cdot \Delta t$$

$$\ell n\ (N_3/N_2) = -C \cdot F(T_3,\ t_3) \cdot \Delta t$$

$$\ddots$$

$$\ell n\ (N_n/N_{n-1}) = -C \cdot F(T_n,\ t_n) \cdot \Delta t$$

Summing the above,

$$\ell n\ (N_n/N_O) = -C \cdot \sum_{i=1}^{n} F(T_i,\ t_i) \cdot \Delta t$$

Hence

$$\ell n\ (N/N_O) = -\lim_{\Delta t \to 0} C \cdot \sum_{i=1}^{\infty} F(T_i,\ t_i) \cdot \Delta t$$

$$= -C \int_{o}^{t} F(T(t),\ d)dt \qquad (3a)$$

which is Eq (3a).

REFERENCES

1. Aiba, S., Humphrey, A. E., and Millis, Nancy F., *Biochemical Engineering*, 2nd ed., Academic Press, 1973.

2. "Overview – Symposia in Food Science and Technology," *Food Technology*, 32, No. 3, March 1978.

3. Wang, C. P. and Brynjolfsson, A., "Heat Transfer and the Killing of Bacteria during Thermal Sterilization of Meat Rolls," *paper 78–WA/HT–56, Winter Annual Meeting of the American Society of Mechanical Engineers* in San Francisco, Dec. 10–15, 1978.

4. Carslow, H. S., and Jaeger, J. C., *Conduction of Heat in Solids,* 2nd ed., Oxford University Press, 1959.

5. Schneider, P. J., *Conduction Heat Transfer,* Addison-Welsly Publishing Co., 1955.

6. Pflug, I. J., and Odlaug, T. E., "A Review of z and F Values Used to Ensure the Safety of Low-Acid Canned Food," *Food Technology*, 32, No. 6, June 1978, pp. 63–67.

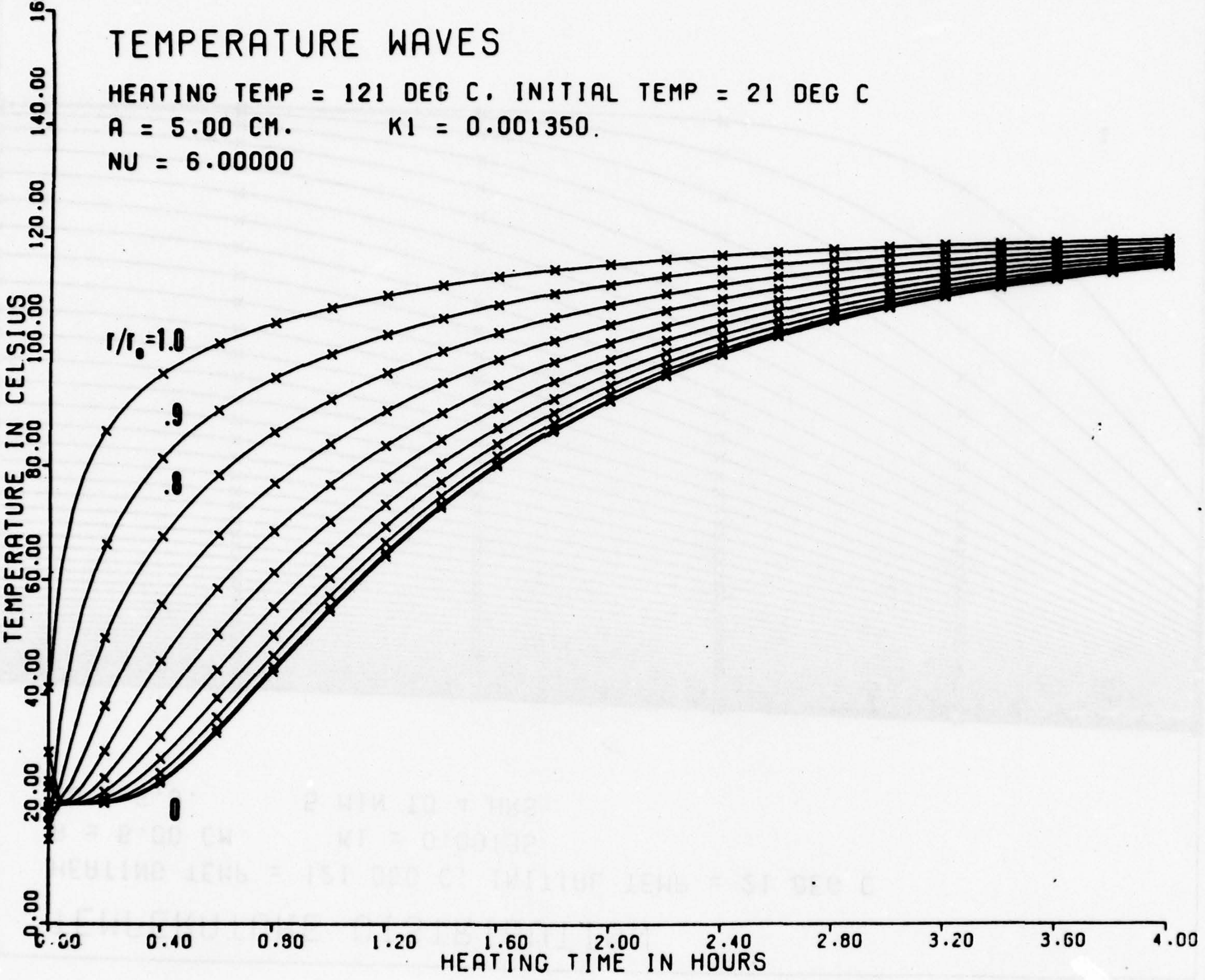


Figure 1.

Computer plot of temperature waves in a long meat roll at radial distances r = 0, 0.1a, 0.2a .... 1.0a, starting from the lowest curves. The crosses are for the purposes of providing the 0.2 hr time marks. Kl is the thermal diffusivity κ, and NU is the effective conduction Nusselt number ν or Biot number. $r_0 \equiv a$.

Computer plot of temperature waves in a long meat roll at radial distances r = 0, 0.1a, 0.2a .... 1.0a, starting from the lowest curves. The crosses are for the purposes of providing the 0.2 hr time marks. KI is the thermal diffusivity κ, and NU is the effective conduction Nusselt number ν or Biot number. $r_o \equiv a$.

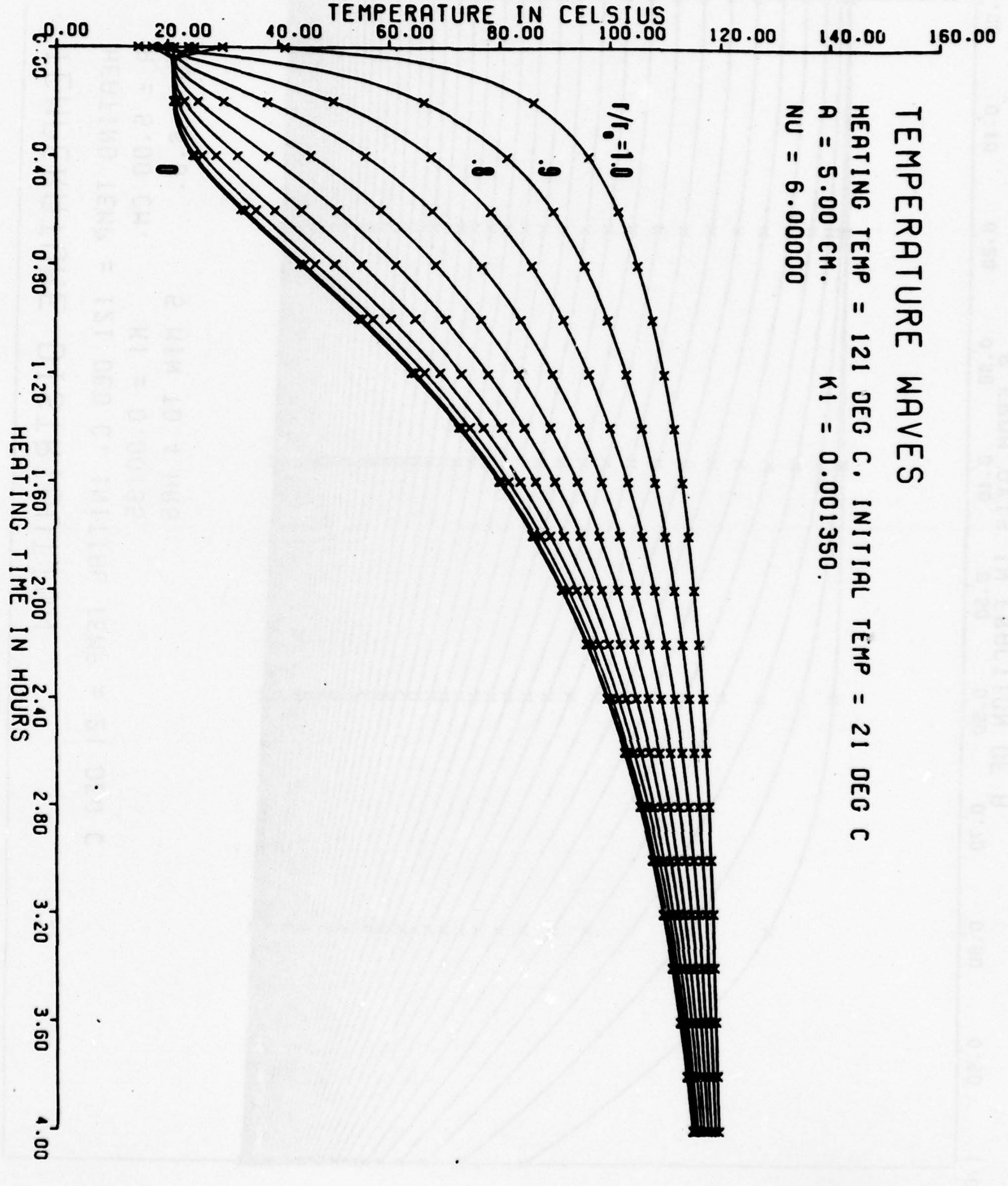


Figure 1.

Computer plot of the temperature distribution in a long meat roll at 5 minutes interval, starting from the lowest curve. The crosses are for the purposes of providing the radial distance markings. KI is the thermal diffusivity κ, and NU is the effective conduction Nusselt number ν or Biot number.

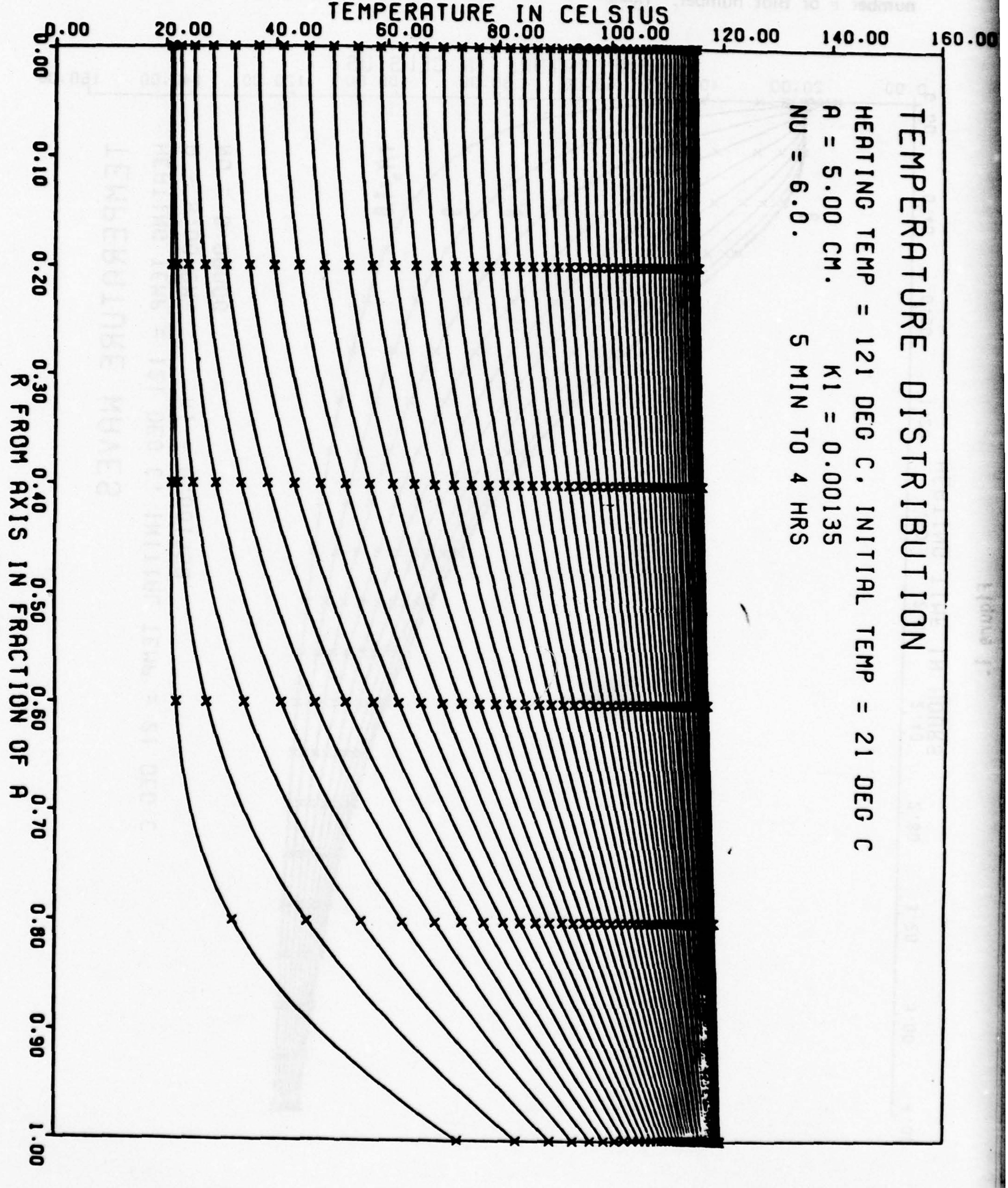
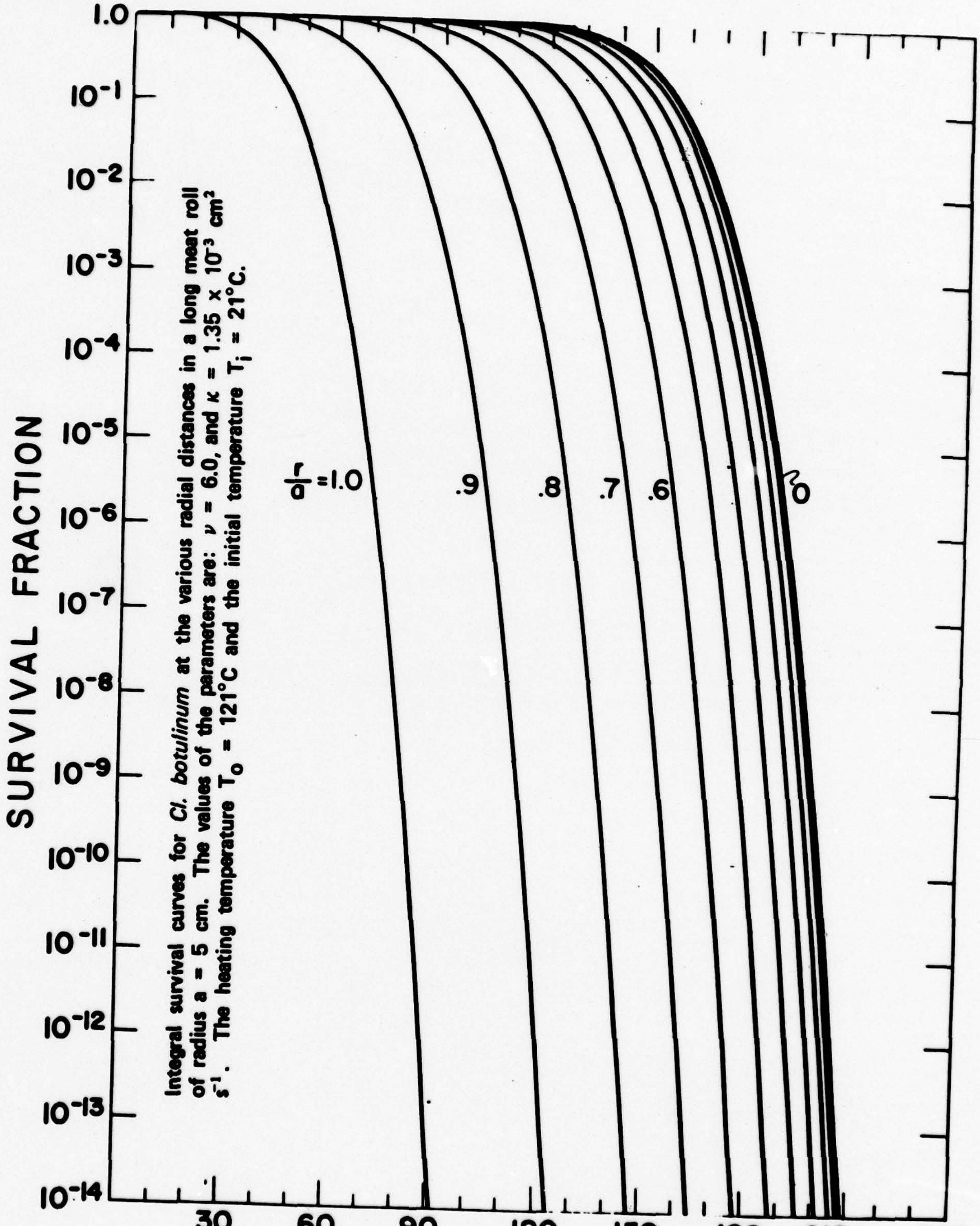


Figure 2.



Integral survival curves for *Cl. botulinum* at the various radial distances in a long meat roll of radius a = 5 cm. The values of the parameters are: ν = 6.0, and κ = 1.35 x 10^{-3} cm^2 s^{-1}. The heating temperature T_0 = 121°C and the initial temperature T_i = 21°C.

Figure 3

NASTRAN ANALYSIS OF A PHOTOPLASTIC MODEL FOR SLIDING BREECH MECHANISM

P. C. T. Chen and G. P. O'Hara
U.S. Army Armament Research and Development Command
Benet Weapons Laboratory
Watervliet Arsenal, Watervliet, NY 12189

ABSTRACT. The piece-wise linear analysis option of the NASTRAN code was used to analyze a photoplastic model for sliding breech mechanism. A two-dimensional finite element representation for the breech ring was chosen and the material was made of polycarbonate resin. The aluminum block was regarded as rigid and the width of contact was assumed to remain unchanged during loading. The displacements and stresses in the breech ring were obtained for loading in the elastic range and then in the plastic range by using an incremental approach. The maximum tensile and compressive stresses as well as the residual stresses after complete unloading were calculated.

1. INTRODUCTION. In guns with a sliding breechblock mechanism, breech ring failures have been observed originating from the lower fillet in the vicinity of the contact region. This observation indicates that high tensile stress produced at the fillet is responsible for the failure. There have been a considerable amount of experimental work done on weapon breeches of this type [1-3]. In order to produce beneficial residual stresses and to extend the fatigue life, an exploratory study on the autofrettage of breech mechanism was initiated. A photoplastic model made of aluminum block and polycarbonate ring was designed [4]. The maximum fillet stress was determined for an elastic load as well as an elastic-plastic load. Residual stress resulting from complete unloading was calculated.

The paper describes a numerical investigation of the photoplastic model. A two-dimensional finite element representation for the breech ring is chosen and the aluminum block is regarded as rigid. The width of contact is assumed to remain unchanged during loading. The piecewise linear analysis option of the NASTRAN code is used to analyze this problem [5]. The displacements and stresses in the breech ring are obtained for loading in the elastic range and then in the plastic range by using an incremental approach. The maximum tensile and compressive stresses as well as the residual stresses after complete unloading are calculated.

2. MODEL AND LOADING. A two-dimensional photoplastic model of the meridan section of a sliding breech mechanism is shown in Figure 1. The breechblock was made of aluminum and the load was applied through a pin at the top of the block by means of dead weight. The breech ring was made of 0.12 inch thick LEXAN plate and the top of the ring was fixed. LEXAN is the trade name for polycarbonate resin. The uniaxial stress-strain curve for LEXAN is shown in Figure 2. This material has a Poisson's ratio of 0.38 in the elastic state and a limiting value of 0.5 in the plastic state. The stress-strain (σ-ε) curve for LEXAN can also be described by the modified Ramberg-Osgood equation in the following form

$$E\varepsilon/\sigma_B = \sigma/\sigma_B + (3/7)(\sigma/\sigma_B)^n \quad \text{for } \sigma \geq \sigma_A \tag{1}$$

where the values of four parameters are

$$E = 325 \text{ Ksi}, \quad n = 11.5, \quad \sigma_A = 6.2 \text{ Ksi}, \quad \sigma_B = 8.7 \text{ Ksi}.$$

Figure 3 shows a finite element representation for one half of the breech ring. The other half is not needed because of symmetry. There are 224 grids and 189 quadrilateral elements in this model. The grids 1 through 8 are constrained in x-direction only while grids 233 through 240 are held fixed. The top portion of the breech ring is omitted because this is believed to have little effect on the maximum stress information near the lower fillet. In fact this belief has been confirmed by obtaining the elastic solution for another finite element model with additional 70 quadrilateral elements in the top portion. The differences between these two models for the maximum tensile and compressive stresses are 1.3%, 3.3%, respectively. The aluminum block is regarded as rigid and the load is transmitted to the ring through contact. Initially the block is in full contact with the ring. As load increases, a gap develops in the central portion. The gap between the block and the ring under full load of 572 pounds was measured and the results are 0.005, 0.004, 0.003, 0.002, 0.0015, 0.0005 inch at a distance of 0.995, 1.125, 1.325, 1.505, 1.625, 1.775 inch, respectively, from the center. NASTRAN program in its present form cannot be used to determine the width of contact and the force distribution as functions of loading. In this numerical investigation, several contact conditions are chosen and assumed to remain unchanged during loading.

3. ELASTIC SOLUTION. Since the width of contact and the force of distribution are not to be determined as functions of loadings, eight contact conditions under different prescribed displacements or forces are tried. The displacements and stresses in the elastic range for all cases are obtained. Three cases are under uniform prescribed displacements; case 1 at nodes (33,41,49,57), case 2 at nodes (49,57,65) and case 3 at nodes (57,65,81). Five cases are under prescribed forces; case 4 (20,50, 30 lbs.) at nodes (49,57,65), case 5 (63.13,36.87 lbs.) at nodes (57,65), case 6 (100 lbs.) at node 65, case 7 (50,50 lbs.) at nodes (65,81), case 8 (12.5,25,25,25,12.5 lbs.) at nodes (57,65,81,89,97). It is interesting to

to find out that the location of the maximum tensile stress is in element 155 for all cases and agrees very well with the experimental observation. The constraint forces for the three cases under prescribed displacements are calculated. If the total contact force F is 100 pounds, the magnitudes of the maximum tensile stress for 8 cases are 1801, 1847, 1869, 1842, 1840, 1853, 1873, 1916 psi, respectively. It should be noted that the stresses in the NASTRAN program are calculated at the centroid of each element but only one principal stress at the force boundary can be measured. The experimental result for the maximum tensile stress at the fillet is 2222 psi per 100 pounds of load in the elastic range. For the purpose of comparison, the boundary stress is determined by extrapolation using the calculated results for those elements along the radial direction through element 155. This is shown in Figure 4 for case 4. It seems that the numerical result agrees very well with the experimental data in the elastic range of loading. The stresses near the contact area are not measured but the numerical results for the 8 cases indicate that the maximum compressive stress always occurs at the edge of contact area and the maximum octohedral shear stress also occurs at or near this location . The total contact force required to cause incipient plastic deformation is calculated for each case. The values are 223, 211, 200, 241, 206, 137, 192, 293 pounds for cases 1 to 8, respectively.

4. ELASTOPLASTIC SOLUTION. The piecewise linear analysis option of the NASTRAN code is used to obtain elastic-plastic solution. The load or displacement is applied in increments such that the stiffness properties can be assumed to be constant over each increment. For each contact condition under prescribed forces or displacements, the elastic solution is first obtained and then scaled to cause incipient plastic deformation by using Mises yield criterion. This scale factor is used as the first increment. The other increments have to be chosen properly by the user in order to obtain good results at reasonable cost. The size of increments depends upon the desired accuracy, material properties, element size, load level, etc. It is not easy to choose a proper set of increments. The increments can be uniform or nonuniform. It seems that smaller increments should be used as plastic deformation become bigger [6].

Only two of the eight contact conditions considered in the elastic range have been extended into the plastic range. They are case 1 under prescribed displacements and case 4 under prescribed forces. RIGID FORMAT 6 of the NASTRAN Code is used in both cases [5]. However, for problems under prescribed displacements, the DMAP sequence should be slightly modified. The modified instructions (Nos. 60 and 113) are inserted in the executive control deck. A partial list of the input deck for case 1 under prescribed displacement is shown in Table 1. The elements with potential high stress states are regarded as nonlinear. These elements are listed in set 1 and a stress-strain table is associated with them.

All other elements with low stress states are regarded as linear even though their actual properties are decidedly nonlinear at high stress levels. There are 189 quadrilateral membrane elements and 224 grid points. The entry in Field 8 of the CQDMEM element connection card is the angle (α°) of the global coordinate system (X,Y) with respect to the element coordinate system (x,y) instead of the material property orientation angle. The permanent single-point constraints associated with grid points (1 through 8, 233 through 240) are entered in Field 8 of the GRID cards. The maximum contact displacement at grid points 33, 41, 49 and 57 is set as 0.089 inch. This magnitude is to be reached in 15 incremental steps defined by PLFACT card. Since no actual applied forces are involved, a dummy force card together with its related information should be provided. The modified instructions (Nos. 31, 136 and 164) in the executive control deck are needed if the output results are to be saved on auxiliary storage device for post-processing. We store them on tape as data set 11. The complete input deck defines a NASTRAN problem.

The output of the NASTRAN program are the displacements and stresses for each load factors. The information on the strains is not available. The displacements at all grid points have been obtained and the vertical displacements along the boundary points 1, 9, 17, 25, 33, 41, 49, and 57 are used to determine the central gap under various loadings. The central gap calculated for contact force F at 203, 544 pounds and the measured data for F at 572 pounds [4] are shown in Figure 5. The main reason for the discrepancy is due to the assumption that the width of contact is chosen and to remain unchanged during loading. The contact force F is the sum of the vertical components of the constraint forces under contact. The results for F at the first 12 load levels are 203, 265, 324, 353, 382, 411, 439, 467, 490, 512, 533, 544. The NASTRAN program stops at load level 13 with the error message - stiffness matrix singular due to material plasticity. The relation between the contact force and the contact displacement is almost linear as shown in Figures 6, 7 or 8.

The stress outputs of the NASTRAN program are σ_x, σ_y, τ_{xy}, β, σ_1, σ_2, τ_m at the centroid of each elements in set 2 where σ_x, σ_y, τ_{xy} are the stresses in element coordinate system (x,y); β is the principal stress angle; σ_1 and σ_2 are the major and minor principal stresses; and τ_m is the maximum shear. Since our stress results are stored on tape, we can retrieve them to calculate the octahedral shear stress (τ_o), the effective stress (σ_o), the stresses in global coordinate system ($\bar{\sigma}_x$, $\bar{\sigma}_y$ and $\bar{\tau}_{xy}$) and we can also calculate the residual stresses after complete unloading from various stages of loadings. The formulas for the above calculations are

$$\tau_o = (\sqrt{2}/3)\sigma_o = (S_x{}^2 + S_y{}^2 + S_z{}^2 + 2\tau_{xy}{}^2)^{1/2} \ ,$$

$$S_x = (2\sigma_x - \sigma_y)/3 \ , \quad S_y = (2\sigma_y - \sigma_x)/3 \ ,$$

$$S_z = -\ (\sigma_x + \sigma_y)/3 \ , \quad \theta = \alpha - \beta \ ,$$

$$\bar{\sigma}_x = 1/2(\sigma_1 + \sigma_2) + 1/2(\sigma_1 - \sigma_2)\cos 2\theta \ ,$$

$$\bar{\sigma}_y = 1/2(\sigma_1 + \sigma_2) - 1/2(\sigma_1 - \sigma_2)\cos 2\theta$$

$$\bar{\tau}_{xy} = -\ 1/2(\sigma_1 - \sigma_2)\sin 2\theta \ . \qquad (2)$$

We have examined the stresses in all elements and the results show that the maximum tensile and compressive stresses occur in element 155 and 50, respectively. These locations remain unchanged as load increases. Therefore we plot the stresses in these two elements as functions of loading history as shown in Figures 6 and 7. The maximum tensile stress is σ_1 in element 155 and the maximum compressive stress is σ_2 in element 50. At F = 544 pounds, $(\sigma_1,\sigma_2,\sigma_o) = (10750, 1656, 10025)$ psi in element 155 and (-566,-9631,-9361) psi in element 50. Since the NASTRAN program stops at load level 13, it fails to give any information for loading larger than 544 pounds. However, it does show that the stresses in element 155 remain unchanged after the contact force F reaches 533 pounds. The values of σ_1, σ_o for this constant stress state are larger than the allowable limit which is 9576 psi. It seems that the true values should be lower than the calculated. Under the usual assumption that the unloading process is purely elastic, the residual stresses after complete unloading from various stages of loadings are calculated and these results in element 50 and 155 are shown in Figure 8. The minor principal stress σ_2 in element 50 is compressive under loadings and its residual state is always in tension. This is a favorable condition. However, the major principal stress σ_1 in element 155 is tensile under loadings and its residual state is still in tension if unloading from the early stages of loadings. This is undesirable. If the contact force is larger than 533 pounds, then the residual stress in element 155 decreases. Since the NASTRAN program fails for loading larger than 544 pounds, we do not know whether the residual stresses in element 155 will ever become compressive if unloading from the maximum full load of 572 pounds as reported in [4]. The octahedral shear stress τ_o is used to determine the size of the plastic zone under various loading levels. The plastic zone at F = 544 pounds is shown as the dark area in Figure 3.

We have also obtained an elastoplastic solution for the problem under prescribed forces at nodes 49, 57, and 65. The distributions of forces at these three points are assumed to be 20, 50, 30%, respectively. We have applied the forces of F = 240, 300, 350, 400, 450, 500, 524, 548, 572 pounds. The NASTRAN program stops at the last load level with the error message - "stiffness matrix singular due to material plasticity". The results for the stresses in element 50 and 155 as functions of contact forces up to 548 pounds are shown in Figure 9. No experimental results for the stresses are available for comparison in this range of loading. The only experimental data in the plastic range of loading is 9300 psi for the maximum tensile stress under the maximum test load at 572 pounds [4], but the NASTRAN fails at the last load level. It is interesting to observe that when the contact force increased from 524 to 548 pounds, the stresses in element 155 remain unchanged and the values for σ_1, σ_2, σ_0 are 10603, 1562, 9914 psi, respectively. We may make a conjecture that a constant state of stress in element 155 is reached after the contact force reaches 524 pounds. Therefore the maximum tensile stress at the last load level is taken as 10600 psi, which is larger than 9300 psi as reported in [4]. Since the allowable stress is 9576 psi, it seems that the experimental data based on the photoelastic model is more reasonable than the estimated value.

In order to calculate the residual stresses resulting from unloading, it is necessary to know the stresses for elastic loading as well as for various stages of loadings in the plastic range. The numerical and experimental results in the elastic range as shown in Figure 4 indicate that a stress of 2220 psi is produced for a force of 100 pounds. The residual stresses after complete unloading from various stages of loading up to 548 pounds can be calculated and the results will be similar to those shown in Figure 8. The residual stress resulting from complete unloading at the maximum test load is estimated to be 2100 psi in compression and the experimental result is 3400 psi in compression as reported in [4]. It should be noted that our estimated value is based on the assumption that the maximum tensile stress before unloading is 10600 psi. There is a definite need to improve our numerical results for the stresses in the plastic range especially under larger values of loadings. More numerical as well as experimental works are needed before reliable prediction on the residual stresses can be made.

5. CONCLUSIONS AND RECOMMENDATIONS. A numerical study on a photoplastic model for sliding breech mechanism has been made by using NASTRAN program. The location and magnitude for the maximum tensile and compressive stresses have been determined for loading in the elastic as well as elastoplastic range. In the elastic range of loading, the numerical results are in good agreement with the experimental data. In the elastoplastic range of loading, the NASTRAN program can be used only up to a certain load limit. Within this limit, residual stresses resulting from complete unloading

were calculated but no experimental results were available for comparison. Beyond this limit, the values for the maximum tensile stress and the residual stress after complete unloading were estimated and compared with the experimental results at the maximum load level. The comparison is not satisfactory and suggests that further investigation is needed.

It is proposed for further work to develop a numerical method for solving elastoplastic problem under larger values of loadings. This is necessary for the determination of the residual stresses resulting from large plastic deformation. The approach to be undertaken may include using other available codes, modifying the existing ones or developing new computer programs.

REFERENCES

1. T. F. MacLaughlin, "Photoelastic Stress Analysis of Conventional and Serrated Slide Block Breech Designs," Watervliet Arsenal Technical Report WVT-6830, August 1968.

2. G. P. O'Hara, "Photoelastic Stress Analysis of a High Pressure Breech," Watervliet Arsenal Technical Report WVT-7057, December 1970.

3. Y. F. Cheng, "On Maximum Fillet Stresses in Breech Ring," Watervliet Arsenal Technical Report WVT-7255, October 1972.

4. Y. F. Cheng, "Photoplastic Study of Residual Stress in an Overloaded Breech Ring," Technical Report ARLCB-TR-78018, November 1978.

5. "NASTRAN User's Manual," NASA SP-222(01), 1972.

6. P. C. T. Chen, "A Comparison of NASTRAN Code and Exact Solution to an Elastic-Plastic Deformation Problem," Proceedings of the 1978 Army Numerical Analysis and Computer Conference, pp. 261-275, October 1978.

TABLE 1. PARTIAL LIST OF INPUT DECK FOR CASE 1 UNDER PRESCRIBED DISPLACEMENT

```
//M137BR JOB 4225-1001-GRMO,CHEN,CLASS=B,TIME=120
// EXEC NASTRAN
//FT06F001 DD UNIT=(TAPE7,,DEFER),VOL=(,RETAIN,SER=NIGHT),DISP=MOD,
// DSN=OUT1,LABEL=1,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=133,TRTCH=ET)
//FT11F001 DD UNIT=(TAPE9,,DEFER),VOL=SER=CHEN1,DSN=STRESS,LABEL=4
//NS.SYSIN DD *
NASTRAN SYSTEM(31)=4618
ID M137BR,CHEN AND O'HARA
APP DISPLACEMENT
SOL 6,0
ALTER 31
OUTPUT2   EST,EQEXIN,BGPDT,SIL,//C,N,-1/C,N,11/C,N,PLAS  $
ALTER 60,60
GP4      CASECC,GEOM4,EQEXIN,SIL,GPDT/RG,YS1,USET,/V,N,LUSET/
         V,N,MPCF1/V,N,MPCF2/V,N,SINGLE/V,N,OMIT/V,N,REACT/
         V,N,NSKIP/V,N,REPEAT/V,N,NOSET/V,N,NUL/V,N,NOA  $
ALTER 113
SAVE     PLFACT  $
ADD      YS1,/YS/V,N,PLFACT  $
ALTER 136
OUTPUT2   ONLES,,,,//C,N,0/C,N,11/C,N,PLAS  $
ALTER 164
OUTPUT2   OES1,,,,//C,N,0/C,N,11/C,N,PLAS  $
OUTPUT2   ,,,,//C,N,-9/C,N,11/C,N,PLAS  $
ENDALTER
TIME 120
CEND
TITLE = M137 BREECH RING
SUBT = PHOTOPLASTIC MODEL SIMULATION
LABE = POLYCARBONATE MATERIAL PROPERTIES
SET 1 =    36 THRU  40, 43 THRU 47, 50 THRU 54, 113 THRU 117,
          120 THRU 124, 127 THRU 131, 134 THRU 138,141 THRU 145,
          148 THRU 152, 155 THRU 159, 162 THRU 166, 169 THRU 173,
          239 THRU 243, 246 THRU 250
SET 2 = 1 THRU 7,29 THRU 259,302 THRU 308
STRESS = 2
DISP = ALL
SPCF = ALL
LOAD = 101
 SPC = 101
PLCO =101
BEGINBULK
CQDMEM        1      10       2      10       9       1 -2.131
        (THRU)
CQDMEM       56      10      64      72      71      63  0.000
CQDMEM      113     110      66      82      81      65 -1.285
        (THRU)
CQDMEM      155     110     122     130     129     121 -66.473
CQDMEM      156     110     123     131     130     122 -63.787
CQDMEM      157     110     124     132     131     123 -59.674
```

TABLE 1. PARTIAL LIST OF INPUT DECK FOR CASE 1
UNDER PRESCRIBED DISPLACEMENT (CONT'D)

```
CQDMEM       158     110     125     133     132     124 -56.458
CQDMEM       159     110     126     134     133     125 -55.281
CQDMEM       160      10     127     135     134     126 -58.627
CQDMEM       161      10     128     136     135     127 -90.000
          (THRU)
CQDMEM       175      10     144     152     151     143 -90.000
CQDMEM       239     110     146     162     161     145 -76.658
          (THRU)
CQDMEM       308      10     232     240     239     231 -90.000
FORCE   101     1000            1000.0          -1.0
GROSET                                                  3456
GRID           1         0.0     2.1660                         13456
GRID           2         0.0     1.8714                         13456
          (THRU)
GRID           8         0.0     0.0                            13456
GRID           9         0.6333  2.1660
          (THRU)
GRID          72         2.8500  0.0
GRID          81         2.9215  2.1725
          (THRU)
GRID         152         6.1250  2.5660
GRID         161         3.2500  2.6485
GRID         232         6.1250  6.6498
GRID         233         3.2500  7.1660                        123456
GRID         234         3.6607  7.1660                        123456
GRID         235         4.0714  7.1660                        123456
GRID         236         4.4821  7.1660                        123456
GRID         237         4.8929  7.1660                        123456
GRID         238         5.3036  7.1660                        123456
GRID         239         5.7143  7.1660                        123456
GRID         240         6.1250  7.1660                        123456
GRID    1000                                                   123456
MAT1    5       3.25+5          0.38
MAT1    6       3.25+5          0.38
MATS1   6       1000
TABLES1 1000
+TAB1000 0.0    0.0     0.0028  916.    0.0056  1832.   0.0085  2747.
+TAB10010.0113  3663.   0.0138  4554.   0.017   5618.   0.0202  6455.
+TAB10020.026   7470.   0.0362  8585.   0.0407  9178.   0.0655  9576.
+TAB10030.2     9576.    ENDT
PLFACT  101     .344    .45     .55     .6      .65     .7      .75
+PLFA101 .8     .84     .88     .92     .94     .96      .98    1.
PQDMEM  10      5       0.120
PQDMEM  110     6       0.120
SPC     101     33      2       -0.089
SPC     101     41      2       -0.089
SPC     101     49      2       -0.089
SPC     101     57      2       -0.089
ENDDATA
```

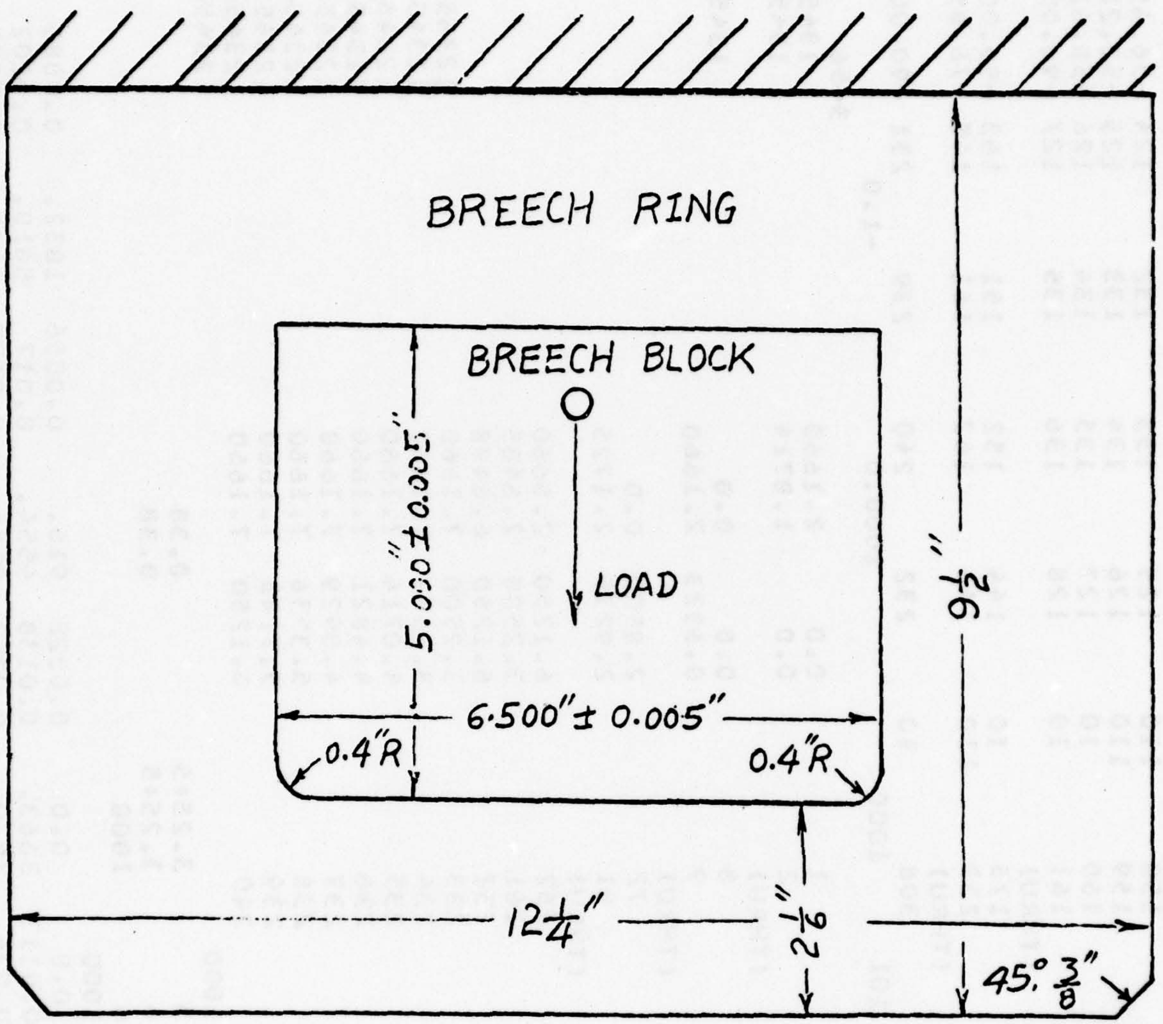


Figure 1. A Photoplastic Model for Breech Mechanism.

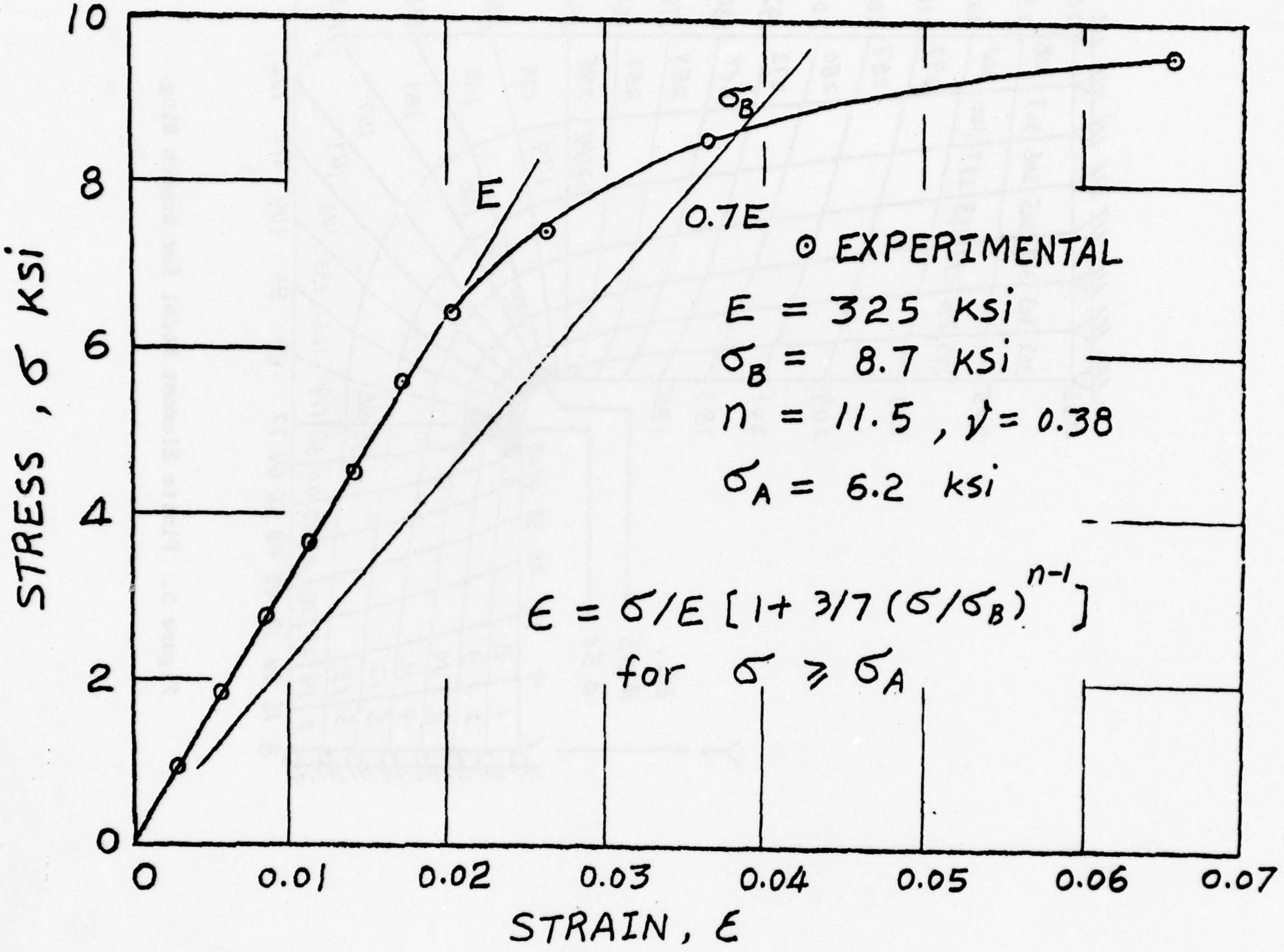


Figure 2. Stress-Strain Curve for LEXAN.

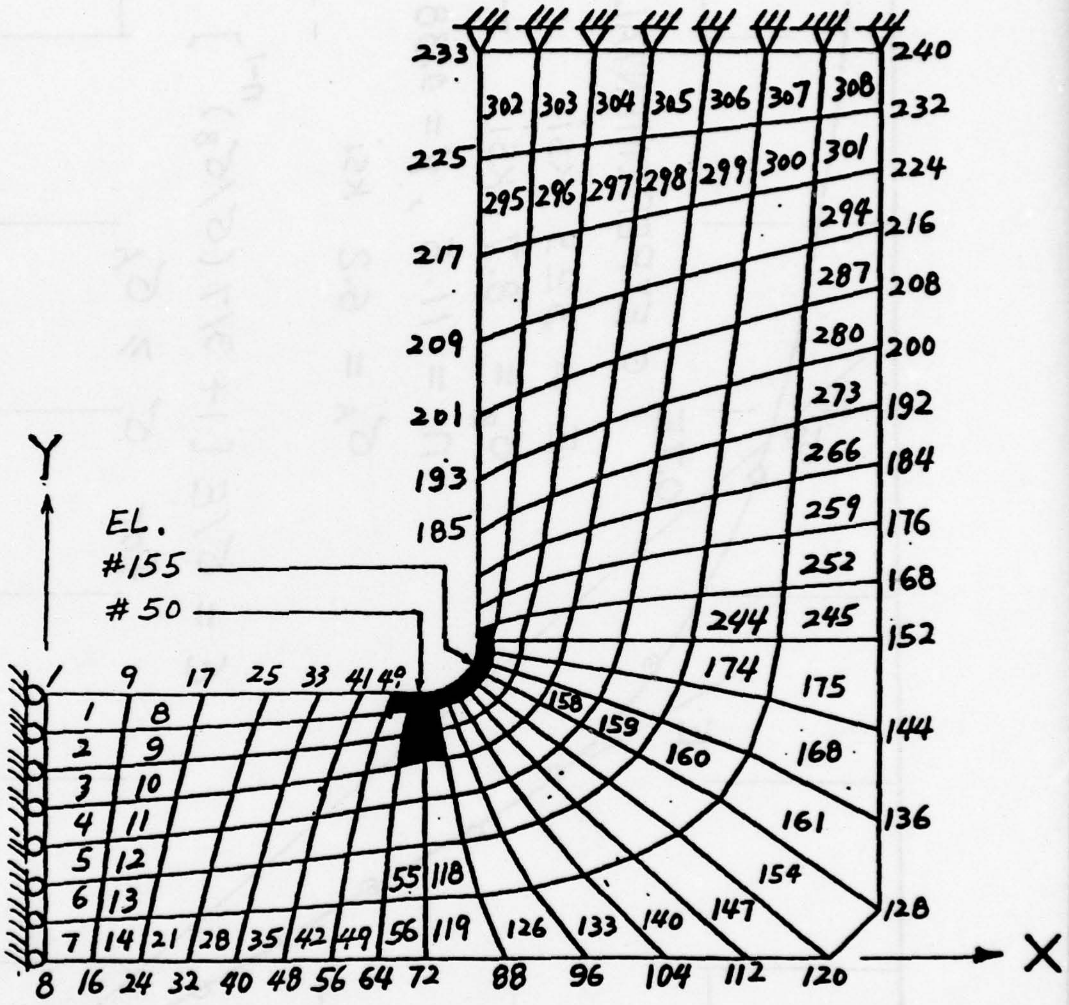


Figure 3. Finite Element Model for Breech Ring.

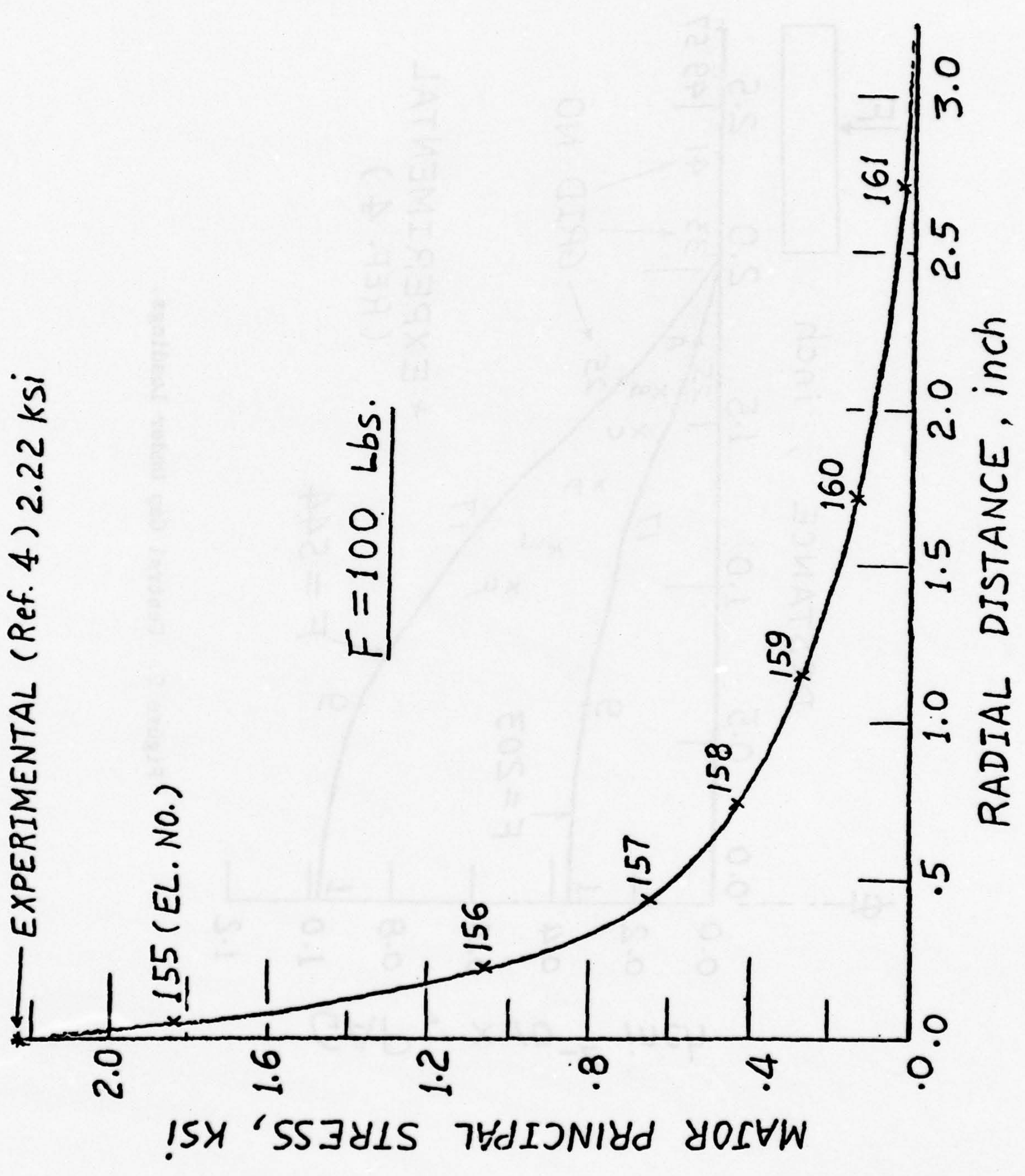


Figure 4. Maximum Tensile Stress for Elastic Loading.

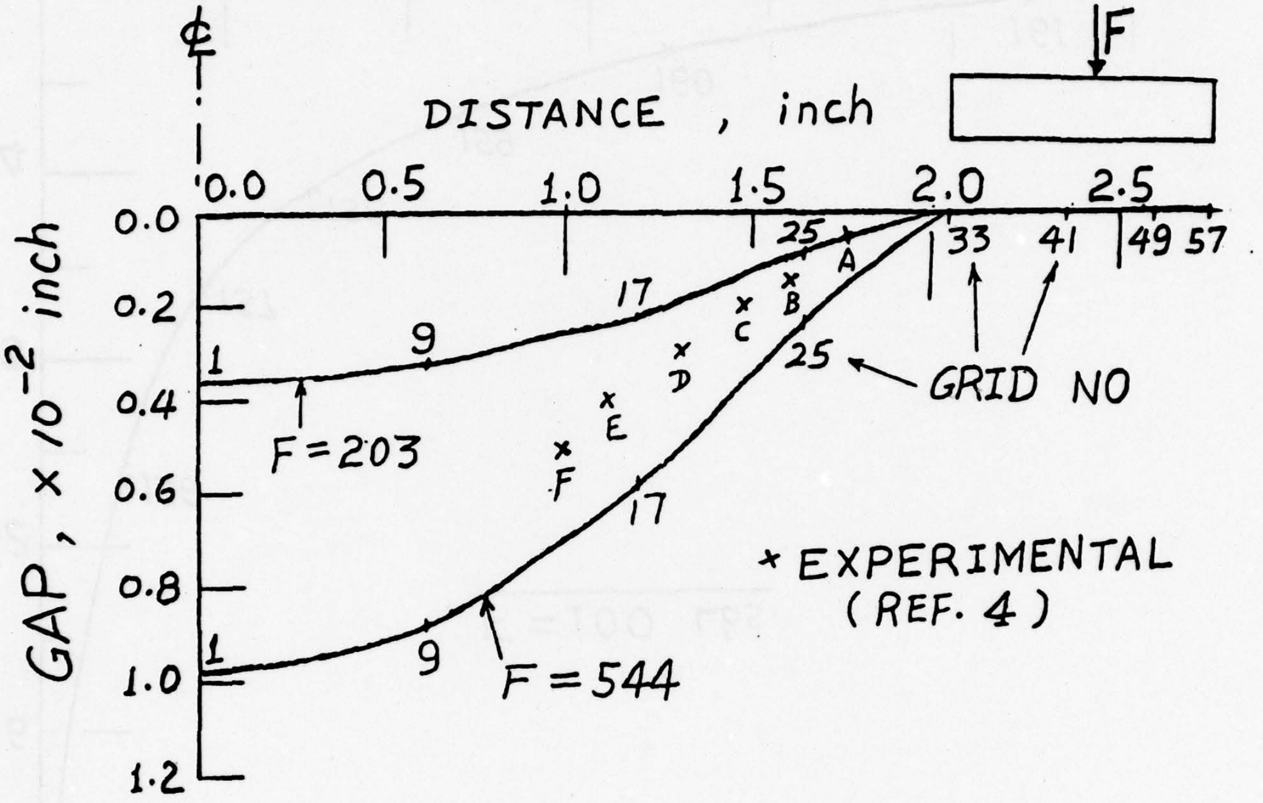


Figure 5. Central Gap Under Loadings.

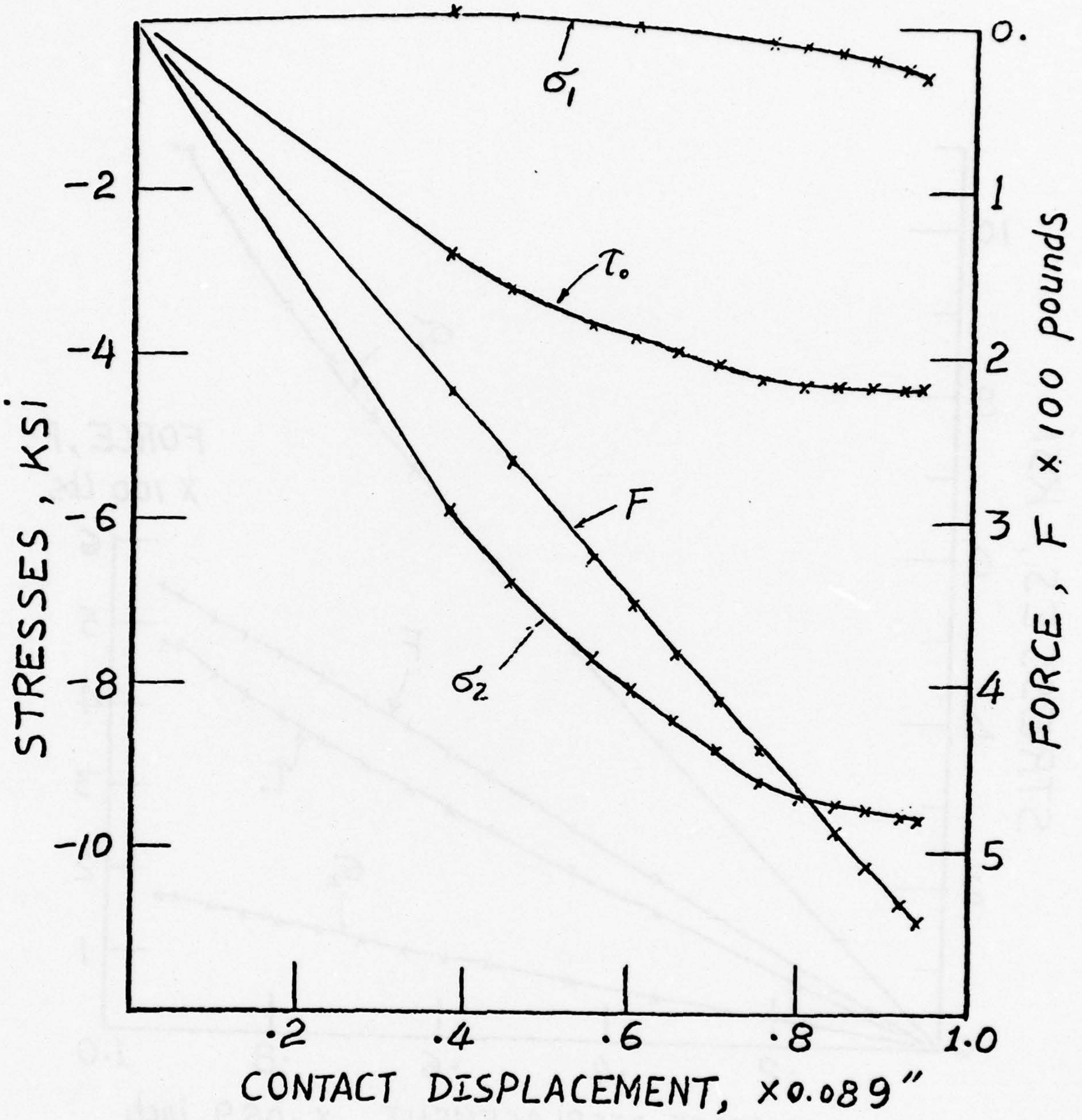


Figure 6. Stresses in Element 50 as Functions of Contact Displacement.

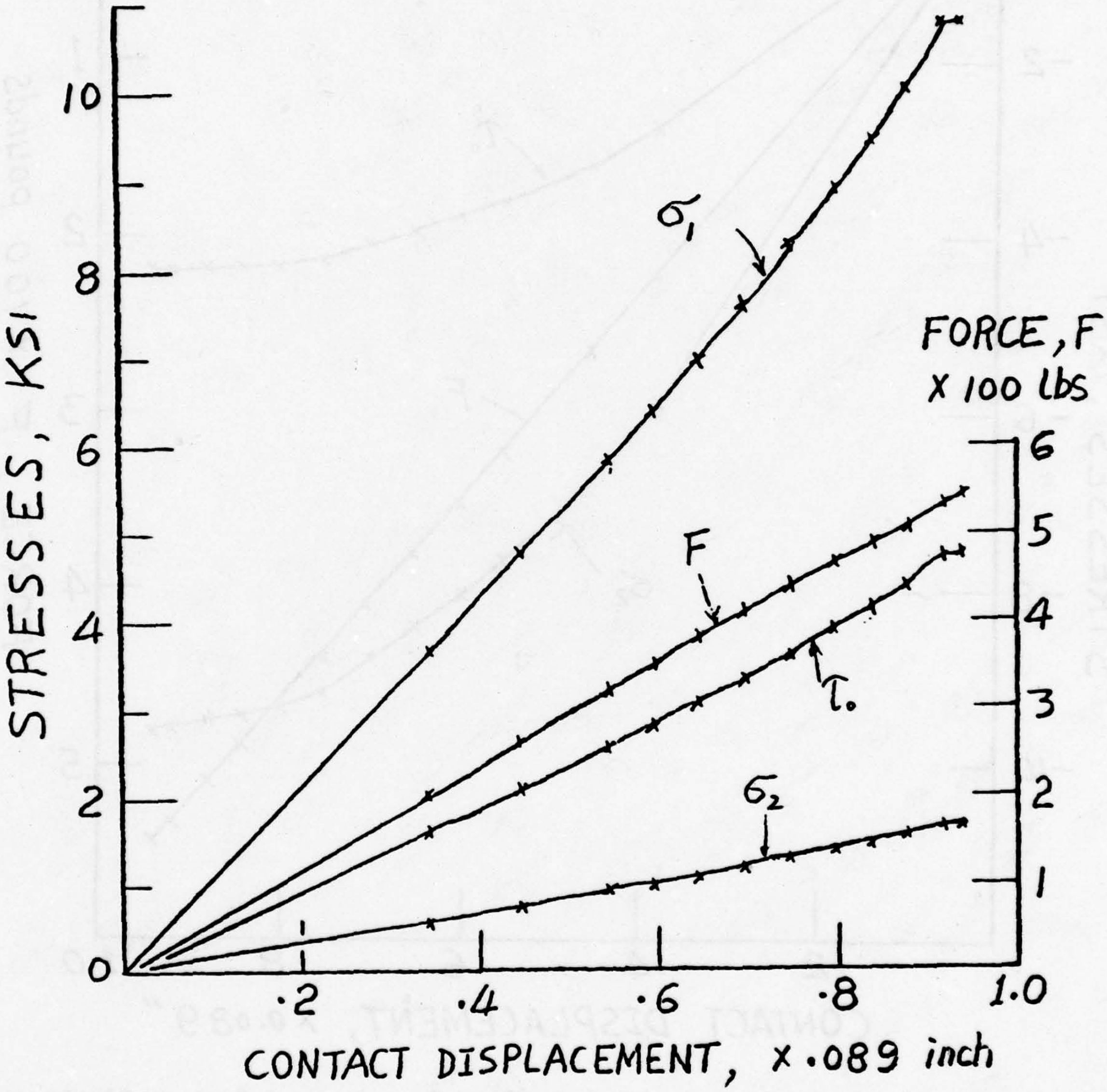


Figure 7. Stresses in Element 155 as Functions of Contact Displacement.

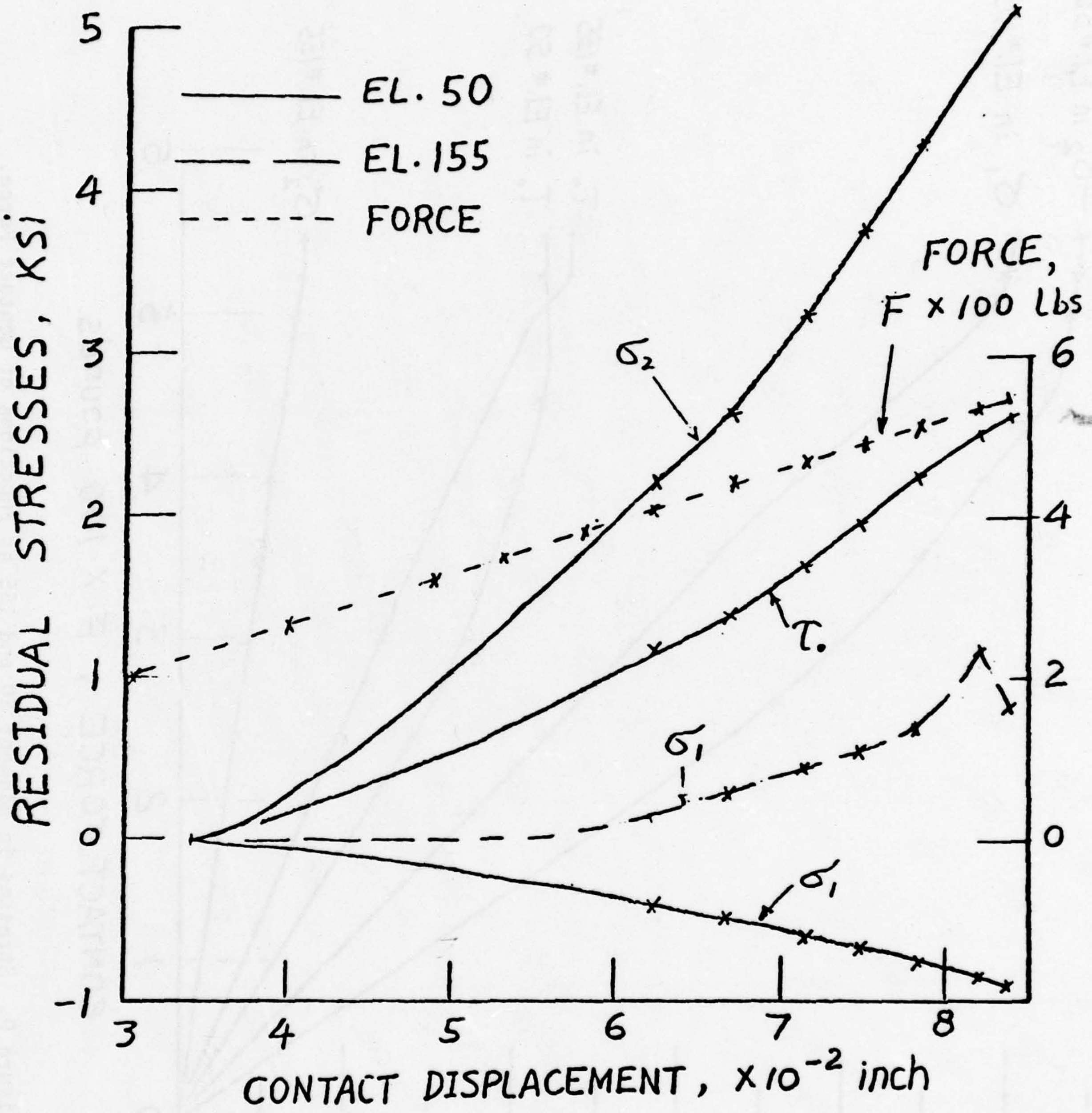


Figure 8. Residual Stresses After Unloading from Various Stages of Loading.

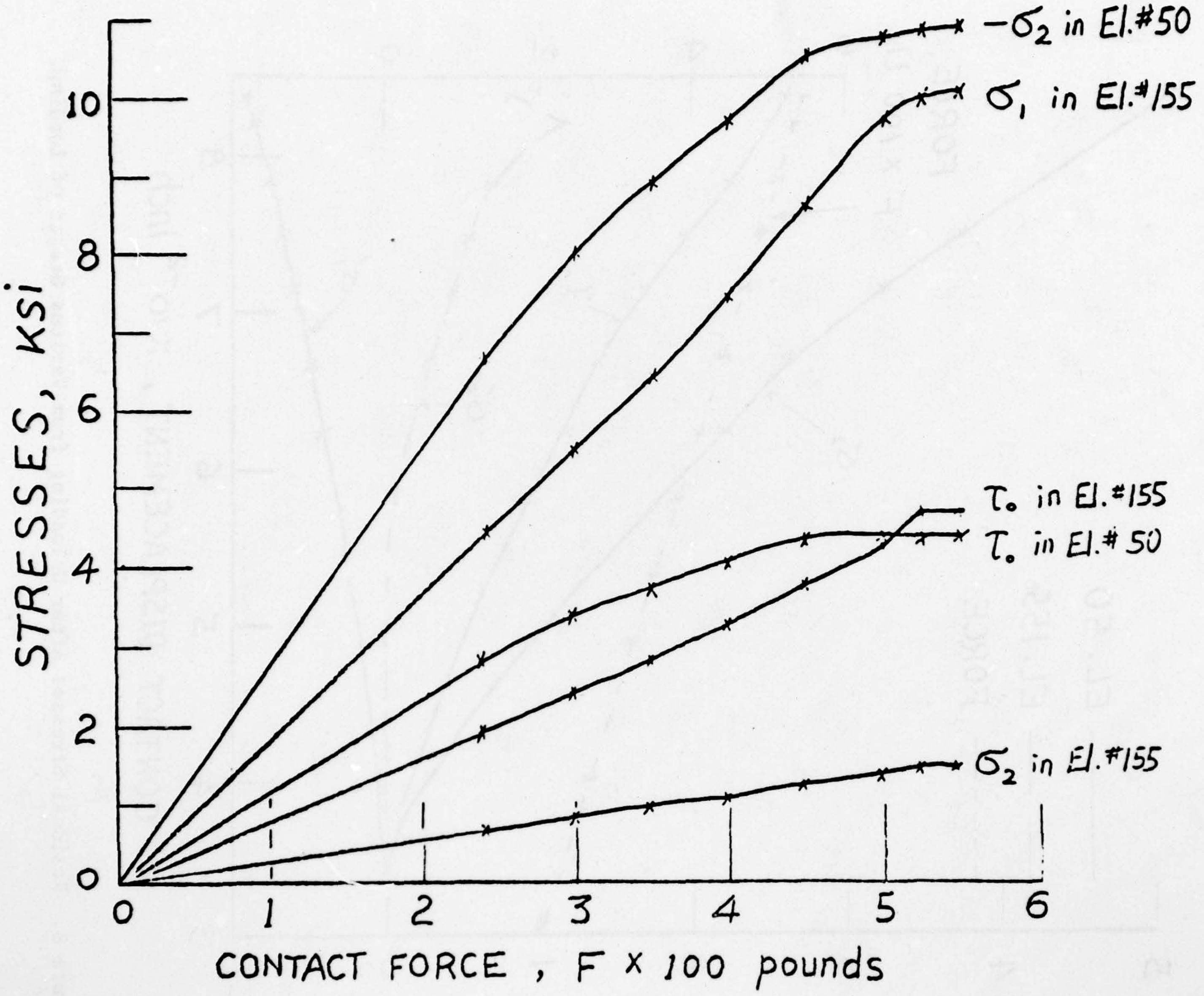


Figure 9. Stresses in Element 50 and 155 as Functions of Contact Force.

A COMPUTER PROGRAM AND APPROXIMATE SOLUTION FORMULATION FOR GUN MOTIONS ANALYSIS

Julian J. Wu
U. S. Army Armament Research and Development Command
Benet Weapons Laboratory, LCWSL
Watervliet Arsenal, Watervliet, NY 12189

ABSTRACT. The purpose of this paper is to describe some of the features associated with a finite element computer program for approximate solutions of a gun dynamics problem. The lateral motion of a gun tube is modeled by an Euler-Bernoulli beam. The difficulties of the problem are due to various complicated loadings and support conditions which can be nonconservative, highly discontinuous and time dependent. The solution formulation for this generally non-self-adjoint problem has been presented in an earlier paper. In terms of finite element discretization, the two-dimensional shape function of spatial and time coordinates is chosen as a product of two one-dimensional shape functions; each for its respective coordinate and both being Hermitian polynomials. The generalized coordinates are then the displacement, slope, velocity and time derivatives of the slope at each node point. The correspondence between local and global generalized coordinates is described. The "stiffness matrices" of spatial and time-effect, contributed by the recoil force, pressure and curvature induced force and the moving mass of a projectile are derived. It is interesting to observe that the strong discontinuities associated with these forces disappear as a result of the smoothing effect of integration in spatial as well as in time coordinates. The present approach to deal with the moving support problem efficiently is also pointed out in this paper. Numerical results of a demonstrative problem are presented.

1. INTRODUCTION. It is our ultimate goal to design a gun system, from which a projectile can hit a target with precision. The technology involved in this task covers that of the interior and exterior ballistics. As it is obvious from the sequence of events, the position, direction and the velocity of a projectile as it leaves the muzzle, consititue the initial conditions for the problem of exterior ballistics. Clearly then, the capability of providing these data under a variety of loading, supporting and detonating conditions is one of the essential links for the complete chain of designing a precision gun system.

This paper reports some of our results in the initial effort on the gun motions analysis. Specifically, these are some of the detailed features related to a computer program for such an analysis. The finite element method in a very general sense has been chosen here due to its proven flexibility and adaptability for a very wide range of situations in terms of geometry, loading and supporting conditions. Even at this early stage, we have observed some unusual features of the problem which render the use of "canned" finite element computer program meaningless. First, there is no satisfactory finite element computer program for initial value programs at the present time other than the methodology developed by this writer using an unconstrained variational formulation [1]. The forces considered in the present study include the moving mass effect of the projectile, the curvature and pressure induced load and the recoil forces which is nonconservative in nature if the support at the breech end is less than rigid. Various elastic supports are considered. In the due course, effects of frictional forces, large deformation and time-dependent support, and other effects will be included. Therefore, it is only sensible to develop right from the beginning a finite element computer program codes, uniquely suitable for the problem on hand, which is capable of handling nonconservative forces, moving mass effects, various supporting conditions including moving supports. In addition, it should also be easily amenable for modifications and inclusions of other effects such as frictional forces, large deformation, and so forth.

The unconstrained variational formulation for approximate solutions of boundary value problems, in conjunction with the finite element descritization has proved to be highly efficient, easy to use and utterly flexible for problems involving nonconservative forces, various support conditions and damping effects [2,3]. Its unique advantage in dealing with initial value problems has also been demonstrated [1]. Therefore it is natural that this finite element unconstrained variational formulation be adapted for the complicated transient problem of gun motions analysis.

The basis of the present formulation for more general cases has been given in a previous paper [4]. The special problem of a uniform gun tube is treated here for demonstrated purposes. The differential, initial and boundary conditions are given in Section 2. This differential equation differs from those used by many other writers by one term. This extra term included in previous papers turned out to be incorrect and the reason is given in Section 3. An unconstrained variational statement which is equivalent to the given governing equation is stated in Section 4. The details of some of the special features on finite element descritization are described in Section 5. Finally some preliminary data which shows the recovery of the initial data are presented in Section 6.

2. GOVERNING EQUATIONS. The motion of a gun tube modeled by the lateral deflection of an Euler-Bernoulli beam is shown in Figure 1. The differential equation in nondimensional form is

$$\begin{aligned} y'''' + (-\bar{P} + g\sin\alpha)[(1-x)y']' + \ddot{y} &= -\bar{P}y''H(\bar{x}-x) \\ &- m[\beta^2 t^2 y'' + 2\beta t\dot{y}' + \ddot{y}]\bar{\delta}(\bar{x}-x) \\ &- (gm\cos\alpha)\bar{\delta}(\bar{x}-x) - g\cos\alpha \end{aligned} \qquad (2\text{-}1)$$

with

$$\bar{P} = \pi R^2 p$$

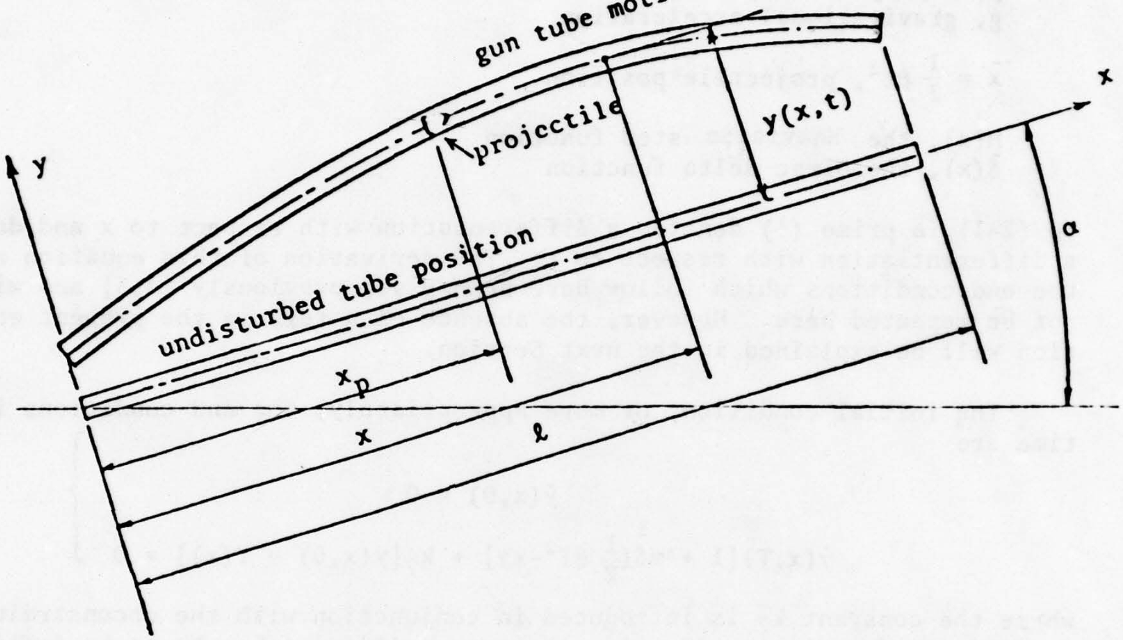


FIGURE 1. A Schematic Drawing of the Problem Configuration.

where

$y = y(x,t)$, the tube deflection
x, spatial axis along the tube's length, $0\le x\le 1$
t, time axis, $0\le t\le T$, T is the time limit of interest
α, elevation angle
m, projectile mass
β, acceleration of the projectile, assumed to be constant
p, bore pressure, assumed to be constant
g, gravitational acceleration

$\bar{x} = \frac{1}{2}\beta t^2$, projectile position

$H(x)$, the Heaviside step function
$\bar{\delta}(x)$, the Dirac delta function

In (2-1), a prime (') denotes a differentation with respect to x and dot (˙), a differentiation with respect to t. The derivation of this equation and the end conditions which follow have been given previously [4,5] and will not be repeated here. However, the absence of a term in the present equation will be explained in the next Section.

The initial condition, or more appropriately, the end conditions in time are

$$\left.\begin{aligned} \dot{y}(x,0) &= 0 \\ \dot{y}(x,T)[1 + m\bar{\delta}(\tfrac{1}{2}\beta T^2 - x)] + k_7[y(x,0) - Y(x)] &= 0 \end{aligned}\right\} \qquad (2\text{-}2)$$

where the constant k_7 is introduced in conjunction with the unconstrained variational formulation (Section 4) so that if one takes k_7 to be infinite, the initial displacement $y(x,0)$ is forced to be identical to the prescribed shape $Y(x)$. Based on similar reasonings the boundary conditions have been shown to be the following.

$$\left.\begin{aligned} & y''(0,t) - k_2 y'(o,t) = 0 \\ & y''(1,t) + k_4 y'(1,t) = 0 \\ \text{and} \quad & \\ & y'''(0,t) + k_1 y(0,t) \\ & \quad + (-\bar{P} + g\cos\alpha)y'(0,t) + \bar{P}y'(0,t)H(\tfrac{1}{2}\beta t^2) \\ & \quad + m\beta^2 y'(0,t)\bar{\delta}(\tfrac{1}{2}\beta t^2) = 0 \\ & y'''(1,t) - k_3(1,t) + \bar{P}y'(1,t)H(\tfrac{1}{2}\beta t^2 - 1) \\ & \quad + m\beta^2 y'(1,t)\bar{\delta}(\tfrac{1}{2}\beta t^2 - 1) = 0 \end{aligned}\right\} \qquad (2\text{-}3)$$

where k_i, i = 1,2,3,4, are the appropriate elastic spring constants at the supports.

3. AN ERROR IN PREVIOUS LITERATURE*. As we have already mentioned that there should be a correction made to the force terms due to the projectile's moving mass. This correction will be explained in the present section.

The geometry of a concentrated mass moving on a "beam" structure is illustrated in Figure 2. In this figure, y(x,t) is the beam deflection. The position vector $\underset{\sim}{r}$ of the moving mass m can be written as

$$\underset{\sim}{r} = \xi(t)\underset{\sim}{i} + y(\xi(t),t)\underset{\sim}{j} \tag{3-1}$$

where i,j are the unit vectors in s and y directions respectively; $\xi(t)$ denotes the position of m in x direction. It is easy to derive the velocity and acceleration of $\underset{\sim}{r}$.

$$\left.\begin{aligned} \underset{\sim}{v} &= \frac{d\underset{\sim}{r}}{dt} = \dot{\xi}(t)\underset{\sim}{i} + [\dot{\xi}(t)y'(\xi(t),t) + \frac{\partial y}{dt}]\underset{\sim}{j} \\ \underset{\sim}{a} &= \frac{d\underset{\sim}{v}}{dt} = \ddot{\xi}(t)\underset{\sim}{i} + [\ddot{\xi}(t)y'(\xi,t) + \dot{\xi}^2(t)y''(\xi,t) \\ &\qquad +2\dot{\xi}(t)\frac{\partial^2 y}{\partial\xi\partial t} + \frac{\partial^2 y}{\partial t^2}]\underset{\sim}{j} \end{aligned}\right\} \tag{3-2}$$

In deriving (3-2), it should be observed that

$$\begin{aligned} \frac{d}{dt}y(\xi(t),t) &= \frac{\partial y}{\partial\xi}\frac{d\xi}{\partial t} + \frac{\partial y}{\partial t} \\ &= \dot{\xi}\frac{\partial y}{\partial\xi} + \frac{\partial y}{\partial t} = \dot{\xi}\,y' + \frac{\partial y}{\partial t} \end{aligned} \tag{3-3}$$

*This error was first observed by T. E. Simkins of Benet Weapons Laboratory at the Second U.S. Army Symposium on Gun Dynamics, 19-22 September 1978, Rensselaerville, New York.

Since $\underset{\sim}{a}$ is the acceleration of the projectile mass m, the force acting on it must be

$$\left.\begin{aligned} \underset{\sim}{F} &= m\underset{\sim}{a} \\ \text{and} \qquad F_x &= ma_x = m\ddot{\xi}(t) \\ F_y &= ma_y \\ &= m\{\ddot{\xi}y' + \dot{\xi}^2 y'' + 2\dot{\xi}\dot{y}' + \frac{\partial^2 y}{\partial t^2}\} \end{aligned}\right\} \qquad (3\text{-}4)$$

It appears that most of the previous investigators have accomplished this much. Now comes the important step. Since we are interested in the force imparted on the beam by the moving mass m. This force, from Figure 2, is clearly not due to F_y alone, rather is the sum of the components of F_x and F_y in the normal direction of the beam. Denote this sum by F_m, one has

$$F_m = F_y \cos\theta - F_x \sin\theta \qquad (3\text{-}5)$$

But, for small deflection, θ is small so that

$$\left.\begin{aligned} \cos\theta &\cong 1 \\ \sin\theta &\cong \tan\theta \cong \theta = \frac{\partial y}{\partial \xi} = y' \end{aligned}\right\} \qquad (3\text{-}6)$$

Substitute (3-6) into (3-5), one obtains

$$\left.\begin{aligned} F_m &= F_y - m\ddot{\xi}y' \\ \text{or} \qquad F_m &= m\{\dot{\xi}^2 y'' + 2\dot{\xi}\frac{\partial y'}{\partial t} + \frac{\partial^2 y}{\partial t^2}\} \end{aligned}\right\} \qquad (3\text{-}7)$$

Eq. (3-7) is the correct expression for the force imparted on a carrying structure due to a moving mass. This structure can be a rail, a bridge or as in the present case a gun tube. It is somewhat surprising to find that the expression for F_y of Eq. (3-4) rather than F_m of Eq. (3-7) is used in the literature known to this writer (see, for example, [6]).

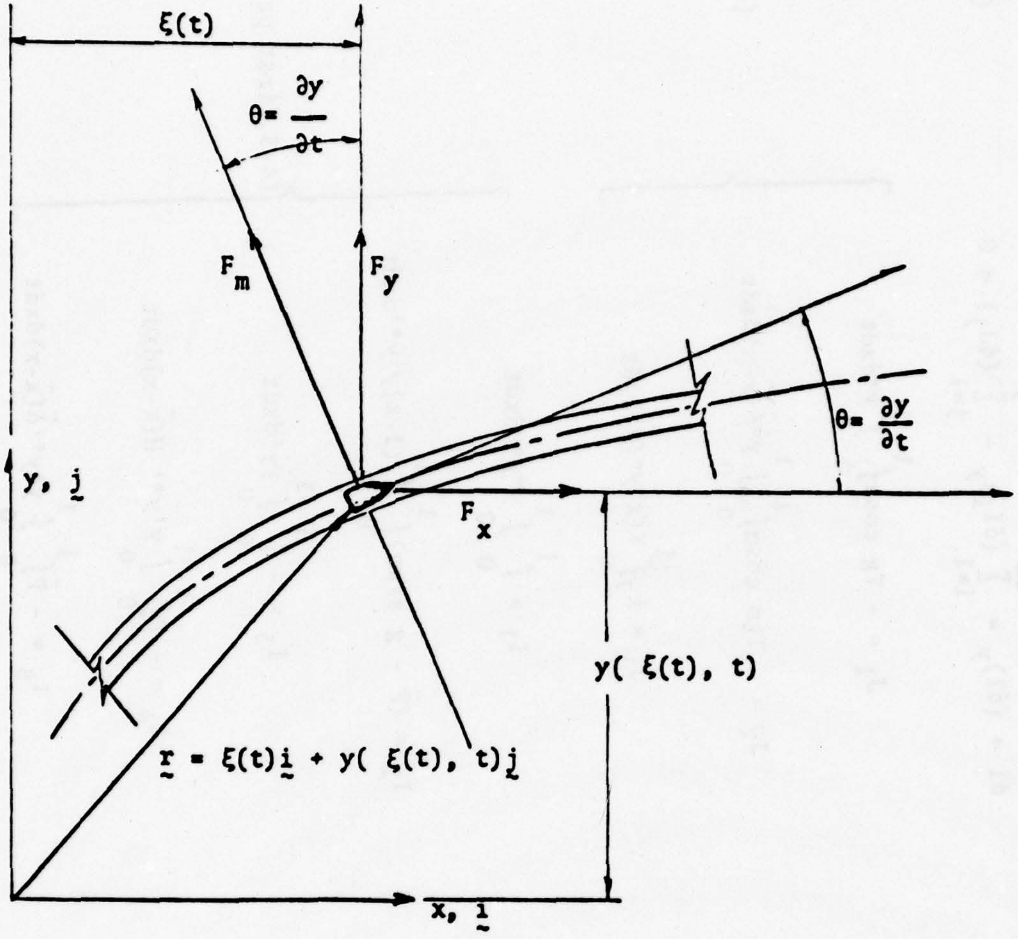

$\xi(t)$
$\theta = \frac{\partial y}{\partial t}$
F_m
F_y
F_x
$\theta = \frac{\partial y}{\partial t}$
$y, \underset{\sim}{j}$
$y(\xi(t), t)$
$\underset{\sim}{r} = \xi(t)\underset{\sim}{i} + y(\xi(t), t)\underset{\sim}{j}$
$x, \underset{\sim}{i}$

FIGURE 2. Forces Acting Between Gun Tube and a Moving Projectile.

4. UNCONSTRAINED VARIATIONAL STATEMENT. Through integrations-by-parts, it is an easy matter to show that the following variational statement is equivalent to the differential equation and ends conditions stated in Section 2.

$$\delta I = (\delta I)_y = \sum_{i=1}^{13} (\delta I_i)_y - \sum_{j=1}^{3} (\delta J_j) = 0 \qquad (4\text{-}1)$$

with

$$\left.\begin{aligned} J_1 &= - Tg \cos\alpha \int_0^1 \int_0^1 y^* dxdt \\ J_2 &= - Tgm \cos\alpha \int_0^1 \int_0^1 y^* \bar{\delta}(\bar{x}-x) dxdt \\ J_3 &= k_7 \int_0^1 Y(x) y^*(x,1) dx \end{aligned}\right\} \qquad (4\text{-}2)$$

and

$$\left.\begin{aligned} I_1 &= \int_0^1 \int_0^1 y'' y^{*''} dxdt \\ I_2 &= (\bar{P} - g \sin\alpha) \int_0^1 \int_0^1 (1-x) y' y^{*'} dxdt \\ I_3 &= - \int_0^1 \int_0^1 \ddot{y} y^* dxdt \\ I_4 &= - \bar{P} \int_0^1 \int_0^1 y' y^{*'} \, H(\bar{x}-x) dxdt \\ I_5 &= - \bar{P} \int_0^1 \int_0^1 y' y^{*'} \bar{\delta}(\bar{x}-x) dxdt \end{aligned}\right\} (4\text{-}3, \text{first part})$$

$$\left.\begin{aligned}
I_6 &= -\frac{m\beta^2}{T}\int_0^1\int_0^1 t^2 y' y^{*\prime}\bar{\delta}(\bar{x}-x)\,dxdt \\
I_7 &= -\frac{m\beta^2}{T}\int_0^1\int_0^1 t\; y' y^{*\prime}\bar{\delta}'(\bar{x}-x)\,dxdt \\
I_8 &= \frac{2m\beta}{T}\int_0^1\int_0^1 t\dot{y}' y^{*}\bar{\delta}(\bar{x}-x)\,dxdt \\
I_9 &= \frac{m\beta}{T}\int_0^1\int_0^1 y' y^{*}\bar{\delta}(\bar{x}-x)\,dxdt \\
I_{10} &= -\frac{m}{T}\int_0^1\int_0^1 \ddot{y} y^{*}\bar{\delta}(\bar{x}-x)\,dxdt \\
I_{11} &= -\frac{m}{T}\int_0^1\int_0^1 \dot{y} y^{*}\dot{\bar{\delta}}(\bar{x}-x)\,dxdt \\
I_{12} &= T\int_0^1 \{k_1 y(0,t)y^*(0,t) + k_2 y'(0,t)y^{*\prime}(0,t) \\
&\quad + k_3 y(1,t)y^*(1,t) + k_4 y'(1,t)y^{*\prime}(1,t) \\
&\quad + k_5 y(x_s,t)y^*(x_s,t) + k_6 y'(x_s,t)y^{*\prime}(x_s,t)\}dt \\
I_{13} &= k_7\int_0^1 y(x,0)y^*(x,1)\,dx
\end{aligned}\right\} \qquad \text{(4-3, second part)}$$

This variational statement will serve as the basis of our finite element solutions.

5. FINITE ELEMENT DISCRETIZATION.

5.1 From Local to Global Coordinates. The purpose of the discretization is to enable one to write the variational statement of Eq. (4-1), which is a functional of continuous functions y and y*, etc., in the form of a matrix equation

$$\delta\underset{\sim}{Y}^{*T}\,\underset{\sim}{K}\,\underset{\sim}{Y} = \delta\underset{\sim}{Y}^{*T}\,\underset{\sim}{F} \qquad \text{(5-1)}$$

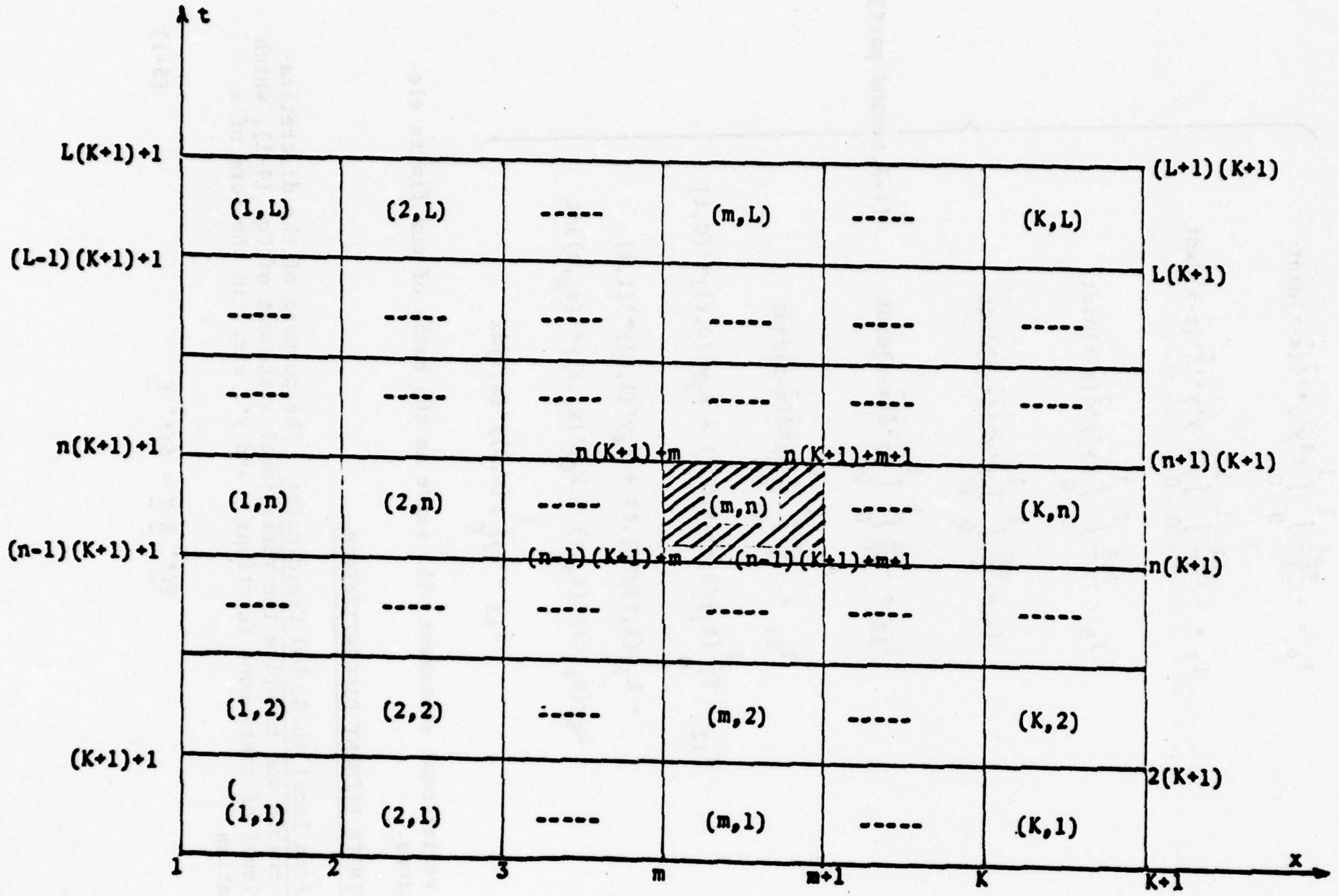


FIGURE 3. Globe Node Scheme for the (m,n)th Element.

where Y, Y* are the "global" generalized coordinates vectors. K is the global "stiffness" matrix, and F the "force" vector. These terminology are patented after the static structural analysis, but they do not necessarily have the physical meanings of those adjectives attached to them. Since the variational statement associated with (5-1) is unconstrained, the equation leads directly to

$$\underset{\sim}{K}\,\underset{\sim}{Y} = \underset{\sim}{F} \tag{5-2}$$

which can be solved for Y if K and F are properly defined. The process by which K and F are assembled and the relation between Y and the desired solution y(x,t) will be described in detail here in this section.

The first step is to write down the expressions in the variational statement in terms of the element variables. A grid scheme of elements is shown in Figure 3. In this figure, the nondimensional length of the gun tube is divided into K equal segments and the time range of interest, into L equal segments. The result is then a set of K x L rectangular elements. In the equations that follow n, the sub- or super-scripts m,n denote the association with the m^{th}, n^{th} segments or the $(m,n)^{th}$ element. Define the relation between the local coordinates (ξ,η) of the $(m,n)^{th}$ element and the global coordinates (x,t) by

$$\begin{aligned} \xi &= \xi^{(m)} = Kx - m + 1 \\ \eta &= \eta^{(n)} = Lt - n + 1 \end{aligned} \tag{5-3}$$

Or

$$\begin{aligned} x &= \frac{1}{K}\,(\xi+m-1) \\ t &= \frac{1}{L}\,(\eta+n-1) \end{aligned} \tag{5-4}$$

One can write, from Eqs. (4-2) and (4-3):

$$\left.\begin{aligned} \delta I_1 &= \sum_{m=1}^{K}\sum_{n=1}^{L} \frac{TK^3}{L} \int_0^1\int_0^1 y''_{(m,n)}\,\delta y^{*''}_{(m,n)}\,d\xi d\eta \\ \delta I_2 &= \sum\sum \frac{T}{L}\,(\bar{P}-g\sin\alpha)\int_0^1\int_0^1 [(K-m+1)-\xi]\,y'_{(m,n)}\,\delta y^{*'}_{(m,n)}\,d\xi d\eta \end{aligned}\right\} \tag{5-5a}$$

$$
\left.
\begin{aligned}
\delta I_3 &= -\sum\sum \frac{L}{TK}\int_0^1\int_0^1 \dot{y}_{(m,n)}\,\delta\dot{y}^*_{(m,n)}\,d\xi d\eta \\
\delta I_4 &= -\sum\sum \frac{\bar{P}TK}{L}\int_0^1\int_0^1 y'_{(m,n)}\,\delta y^{*\prime}_{(m,n)}\,H_{(m,n)}\,(\bar{\xi}-\xi)\,d\xi d\eta \\
\delta I_5 &= \sum\sum \frac{\bar{P}TK}{L}\int_0^1\int_0^1 y'_{(m,n)}\,\delta y^*_{(m,n)}\,\bar{\delta}_{(m,n)}\,(\bar{\xi}-\xi)\,d\xi d\eta \\
\delta I_6 &= -\sum\sum \frac{m\beta^2T^3K^2}{L^3}\int_0^1\int_0^1 [(m-1)^2+2(m-1)\eta+\eta^2]\,y'_{(m,n)}\,\delta y^{*\prime}_{(m,n)}\,\bar{\delta}_{(m,n)}\,(\bar{\xi}-\xi)\,d\xi d\eta \\
\delta I_7 &= -\sum\sum \frac{m\beta^2T^3K^2}{L^3}\int_0^1\int_0^1 [(m-1)^2+2(m-1)\eta+\eta^2]\,y'_{(m,n)}\,\delta y^{*\prime}_{(m,n)}\,\bar{\delta}'_{(m,n)}\,(\bar{\xi}-\xi)\,d\xi d\eta \\
\delta I_8 &= \sum\sum \frac{2m\beta TK}{L}\int_0^1\int_0^1 [(m-1)+\eta]\,\dot{y}'_{(m,n)}\,\delta y^*_{(m,n)}\,\bar{\delta}_{(m,n)}\,(\bar{\xi}-\xi)\,d\xi d\eta \\
\delta I_9 &= \sum\sum \frac{m\beta TK}{L}\int_0^1\int_0^1 y'_{(m,n)}\,\delta y^*_{(m,n)}\,\bar{\delta}_{(m,n)}\,(\bar{\xi}-\xi)\,d\xi d\eta \\
\delta I_{10} &= -\sum\sum \frac{mL}{T}\int_0^1\int_0^1 \dot{y}_{(m,n)}\,\delta\dot{y}^*_{(m,n)}\,\bar{\delta}_{(m,n)}\,(\bar{\xi}-\xi)\,d\xi d\eta \\
\delta I_{11} &= -\sum\sum \frac{mL}{T}\int_0^1\int_0^1 \dot{y}_{(m,n)}\,\delta y^*_{(m,n)}\,\dot{\bar{\delta}}_{(m,n)}\,(\bar{\xi}-\xi)\,d\xi d\eta \\
\delta I_{12} &= \sum_{n=1}^{L} \frac{T}{L}\int_0^1 [k_1 y_{(1,n)}\,\delta y^*_{(1,n)} + k_2K^2 y'_{(1,n)}\,\delta y^{*\prime}_{(1,n)} \\
&\qquad + k_3 y_{(K,n)}\,\delta y^*_{(K,n)} + k_4K^2 y'_{(K,n)}\,\delta y^{*\prime}_{(K,n)}]\,d\eta \\
\delta I_{13} &= \sum_{m=1}^{K} \frac{k_7}{K}\int_0^1 y_{(m,1)}\,\delta y^*_{(m,L)}\,d\xi
\end{aligned}
\right\}\quad (5\text{-}5b)
$$

and

$$\left.\begin{aligned}
\delta J_1 &= -\sum_{m=1}^{K}\sum_{n=1}^{L}\frac{Tg\cos\alpha}{KL}\int_0^1\int_0^1 \delta y^*_{(m,n)}\,d\xi d\eta \\
\delta J_2 &= -\sum\sum\frac{Tg\cos\alpha}{L}\int_0^1\int_0^1 \delta y^*_{(m,n)}\bar{\delta}_{(m,n)}(\bar{\xi}-\xi)\,d\xi d\eta \\
\delta J_3 &= \sum_{m=1}^{K}\frac{k_7}{K}\int_0^1 Y_{(m)}(\xi)\,\delta y^*_{(m,L)}\,d\xi
\end{aligned}\right\} \quad (5\text{-}6)$$

Now, the shape functions are introduced. Let

$$\begin{aligned}
y_{(m,n)}(\xi,\eta) &= \underset{\sim}{a}^T(\xi,\eta)\underset{\sim}{Y}^{(m,n)} \\
y_{(m,n)}(\xi,\eta) &= \underset{\sim}{a}^T(\xi,\eta)\underset{\sim}{Y}^{*(m,n)}
\end{aligned} \quad (5\text{-}7)$$

where $a(\xi,\eta)$ is the shape function vector to be chosen and $Y^{(m,n)}$ is the generalized coordinates vector associated with the $(m,n)^{th}$ element. The choice of $a(\xi,\eta)$ and the meaning of $\underset{\sim}{Y}^{(m,n)}$ will now be given. We shall further write

$$\begin{aligned}
y_{(m,n)}(\xi,\eta) &= \underset{\sim}{a}^T(\xi,\eta)\underset{\sim}{Y}^{(m,n)} = \sum_{k=1}^{16} a_k(\xi,\eta)Y_k^{(m,n)} \\
y^*_{(m,n)}(\xi,\eta) &= \underset{\sim}{a}^T(\xi,\eta)\underset{\sim}{Y}^{(m,n)} = \sum_{k=1}^{16} a_k(\xi,\eta)Y_k^{*(m,n)}
\end{aligned} \quad (5\text{-}8)$$

Here in this paper, we select $a_k(\xi,\eta)$ as

$$a_k(\xi,\eta) = b_i(\xi)b_j(\eta), \quad \begin{matrix} i,j=1,2,3,4 \\ k=1,2,\ldots 16 \end{matrix} \quad (5\text{-}9)$$

where $b_i(\xi)$ or $b_i(\eta)$ are defined by

$$\begin{aligned}
b_1(\xi) &= 1 - 3\xi^2 + 2\xi^3 \\
b_2(\xi) &= \xi - 2\xi^2 + \xi^3 \\
b_3(\xi) &= 3\xi^2 - 2\xi^3 \\
b_4(\xi) &= -\xi^2 + \xi^3
\end{aligned} \quad (5\text{-}10)$$

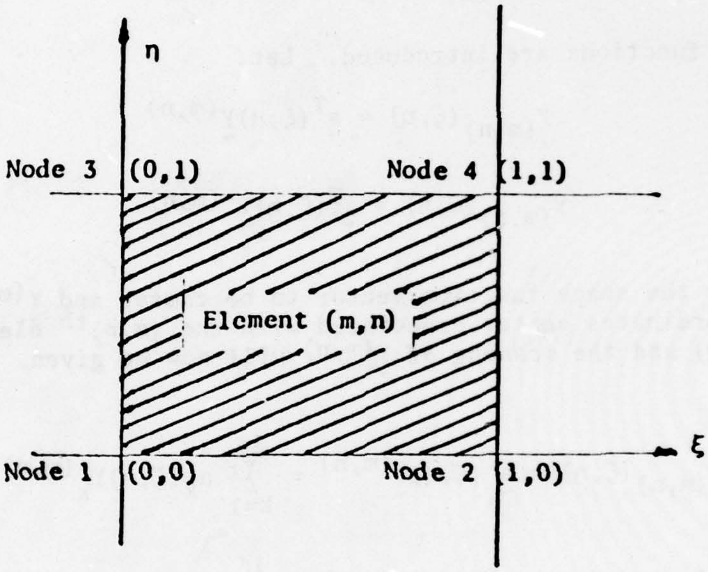


FIGURE 4. Element Node Scheme.

TABLE I. CORRESPONDENCE BETWEEN LOCAL AND GLOBAL NODE NUMBERS

Node Number	
Local	Global
1	(n-1)(K+1)+m
2	(n-1)(K+1)+m+1
3	n(K+1)+m
4	n(K+1)+m+1

Now, the local node scheme is defined in Figure 4 and the correspondence between a local node of the $(m,n)^{th}$ element and a global node is given in Table I.

With Eqs. (5-9) and (5-10) in conjunction of the equation of discretization (5-8), the correspondence between the index k and the pair (i,j) of equation (5-9) and that between $Y_k(m,n)$ and $y_{(m,n)}(\xi,\eta)$ can be easily established. These relations are given in Table II.

TABLE II. CORRESPONDENCE BETWEEN THE INDEX k AND THE PAIR (i,j) OF EQUATION (5-9); AND THAT BETWEEN $Y_{(m,n)k}$ AND $y_{(m,n)}(\xi,\eta)$ AT THE LOCAL NODE POINTS.

k	i	j	$Y_{(m,n)k}$
1	1	1	$Y_1 = y\ (0,0)$
2	2	1	$Y_2 = y_\xi\ (0,0)$
3	1	2	$Y_3 = y_\eta\ (0,0)$
4	2	2	$Y_4 = y_{\xi\eta}(0,0)$
5	3	1	$Y_5 = y\ (1,0)$
6	4	1	$Y_6 = y_\xi\ (1,0)$
7	3	2	$Y_7 = y_\eta\ (1,0)$
8	4	2	$Y_8 = y_{\xi\eta}(1,0)$
9	1	3	$Y_9 = y\ (0,1)$
10	2	3	$Y_{10} = y_\xi\ (0,1)$
11	1	4	$Y_{11} = y_\eta\ (0,1)$
12	2	4	$Y_{12} = y_{\xi\eta}(0,1)$
13	3	3	$Y_{13} = y\ (1,1)$
14	4	3	$Y_{14} = y_\xi\ (1,1)$
15	3	4	$Y_{15} = y_\eta\ (1,1)$
16	4	4	$Y_{16} = y_{\xi\eta}(1,1)$

In terms of $\underset{\sim}{a}(\xi,\eta)$, Y and Y*, the expressions of δI_i, δJ_i, etc. can be written as

$$\delta I_1 = \sum_{m=1}^{K}\sum_{n=1}^{L} \frac{TK^3}{L}\,\delta \underset{\sim}{Y}^{*T}_{(m,n)} \int_0^1\!\!\int_0^1 \underset{\sim}{a}_{,\xi\xi}\underset{\sim}{a}^T_{,\xi\xi}\, d\xi d\eta\, \underset{\sim}{Y}_{(m,n)}$$

$$\delta I_2 = \sum_{m=1}^{K}\sum_{n=1}^{L} (\bar{P}-g\sin\alpha)\,\frac{T}{L}\,\delta \underset{\sim}{Y}^{*T}_{(m,n)} \int_0^1\!\!\int_0^1 [K-m+1-\xi]\underset{\sim}{a}_{,\xi}\underset{\sim}{a}^T_{,\xi}\, d\xi d\eta\, \underset{\sim}{Y}_{(m,n)}$$

$$\delta I_3 = \sum_{m=1}^{K}\sum_{n=1}^{L} \left(-\frac{L}{TK}\right)\delta \underset{\sim}{Y}^{*T}_{(m,n)} \int_0^1\!\!\int_0^1 \underset{\sim}{a}_{,\eta}\underset{\sim}{a}^T_{,\eta}\, d\xi d\eta\, \underset{\sim}{Y}_{(m,n)}$$

$$\delta I_4 = \sum\sum \left(-\frac{TK\bar{P}}{L}\right)\delta \underset{\sim}{Y}^{*T}_{(m,n)} \int_0^1\!\!\int_0^1 \underset{\sim}{a}_{,\xi}\underset{\sim}{a}^T_{,\xi}\, H_{(m,n)}(\bar{\xi}-\xi)\, d\xi d\eta\, \underset{\sim}{Y}_{(m,n)}$$

$$\delta I_5 = \sum\sum \left(\frac{TK\bar{P}}{L}\right)\delta \underset{\sim}{Y}^{*T}_{(m,n)} \int_0^1\!\!\int_0^1 \underset{\sim}{a}\,\underset{\sim}{a}^T_{,\xi}\, \bar{\delta}_{(m,n)}(\bar{\xi}-\xi)\, d\xi d\eta\, \underset{\sim}{Y}_{(m,n)}$$

$$\delta I_6 = \sum\sum \left(-\frac{m\beta^2T^3K^2}{L^3}\right)\delta \underset{\sim}{Y}^{*T}_{(m,n)} \int_0^1\!\!\int_0^1 [(m-1)^2+2(m-1)\eta+\eta^2]\,\cdot$$
$$\cdot\, \underset{\sim}{a}_{,\xi}\underset{\sim}{a}^T_{,\xi}\, \bar{\delta}_{(m,n)}(\bar{\xi}-\xi)\, d\xi d\eta\, \underset{\sim}{Y}_{(m,n)}$$

$$\delta I_7 = \sum\sum \left(-\frac{m\beta^2T^3K^2}{L^3}\right)\delta \underset{\sim}{Y}^{*T}_{(m,n)} \int_0^1\!\!\int_0^1 [(m-1)^2+2(m-1)\eta+\eta^2]\,\cdot$$
$$\cdot\, \underset{\sim}{a}\,\underset{\sim}{a}^T_{,\xi}\, \bar{\delta}'_{(m,n)}(\bar{\xi}-\xi)\, d\xi d\eta\, \underset{\sim}{Y}_{(m,n)}$$

$$\delta I_8 = \sum\sum \left(2m\beta\,\frac{TK}{L}\right)\delta \underset{\sim}{Y}^{*T}_{(m,n)} \int_0^1\!\!\int_0^1 [(m-1)+\eta]\underset{\sim}{a}\,\underset{\sim}{a}_{,\xi\eta}\, \bar{\delta}_{(m,n)}(\bar{\xi}-\xi)\, d\xi d\eta\, \underset{\sim}{Y}_{(m,n)}$$

(5-11,part A)

$$\delta I_9 = \sum\sum \frac{m\beta TK}{L} \delta \underset{\sim}{Y}^{*T}_{(m,n)} \int_0^1 \int_0^1 \underset{\sim}{a}\, \underset{\sim}{a}^T_{,\xi}\, \bar{\delta}_{(m,n)} (\bar{\xi}-\xi) d\xi d\eta\, \underset{\sim}{Y}_{(m,n)}$$

$$\delta I_{10} = \sum\sum (-\frac{mL}{T}) \delta \underset{\sim}{Y}^{*T}_{(m,n)} \int_0^1 \int_0^1 \underset{\sim}{a}_{,\eta} \underset{\sim}{a}^T_{,\eta}\, \bar{\delta}_{(m,n)} (\bar{\xi}-\xi) d\xi d\eta\, \underset{\sim}{Y}_{(m,n)}$$

$$\delta I_{11} = \sum\sum (-\frac{mL}{T}) \delta \underset{\sim}{Y}^{*T}_{(m,n)} \int_0^1 \int_0^1 \underset{\sim}{a}\, \underset{\sim}{a}^T_{,\eta}\, \dot{\bar{\delta}}_{(m,n)} (\bar{\xi}-\xi) d\xi d\eta\, \underset{\sim}{Y}_{(m,n)}$$

$$\delta I_{12} = \sum_{n=1}^{L} (\frac{T}{L}) \{ \delta \underset{\sim}{Y}^{*T}_{(1,n)} [k_1 \int_0^1 \underset{\sim}{a}(0,\eta) \underset{\sim}{a}^T(0,\eta) d\eta$$

$$+ k_2 K^2 \int_0^1 \underset{\sim}{a}_{,\xi}(0,\eta) \underset{\sim}{a}^T_{,\xi}(0,\eta) d\eta] \underset{\sim}{Y}_{(1,n)}$$

$$+ \delta \underset{\sim}{Y}^{*T}_{(K,n)} [k_3 \int_0^1 \underset{\sim}{a}(1,\eta) \underset{\sim}{a}^T(1,\eta) d\eta$$

$$+ k_4 K^2 \int_0^1 \underset{\sim}{a}_{,\xi}(1,\eta) \underset{\sim}{a}^T_{,\xi}(1,\eta) d\eta] \underset{\sim}{Y}_{(K,n)} \}$$

(5-11,part B)

$$\delta I_{13} = \sum_{m=1}^{K} \frac{k_7}{K} \delta \underset{\sim}{Y}^{*}_{(m,L)} \int_0^1 a(\xi,1) a^T(\xi,0) d\xi\, \underset{\sim}{Y}_{(m,1)}$$

Also

$$\delta J_1 = \sum_{m=1}^{K} \sum_{n=1}^{L} (-\frac{Tg\cos\alpha}{KL}) \delta \underset{\sim}{Y}^{*T}_{(m,n)} \int_0^1 \int_0^1 a(\xi,\eta) d\xi d\eta$$

$$\delta J_2 = \sum_{m=1}^{K} \sum_{n=1}^{L} (-\frac{Tgm\cos\alpha}{L}) \delta \underset{\sim}{Y}^{*T}_{(m,n)} \int_0^1 \int_0^1 \underset{\sim}{a}(\xi,\eta) \bar{\delta}_{(m,n)} (\bar{\xi}-\xi) d\xi d\eta$$

$$\delta J_3 = \sum_{m=1}^{K} (\frac{k}{K}) \delta Y^{*T}_{(m,L)} \int_0^1 \int_0^1 Y_{(m)}(\xi) \underset{\sim}{a}(\xi,1) d\xi \qquad (5\text{-}12)$$

With δI_i, δJ_i of the variational statement Eq. (4-1) written in terms of $\underset{\sim}{a}(\xi,\eta)$, $\underset{\sim}{Y}$ and $\delta \underset{\sim}{Y}^*$ as given in Eqs. (5-11) and (5-12), we can now discuss the construction of various element matrices and force vectors.

5.2 Common Element Matrices. A common element matrix is one which is the same for all elements. There are four such matrices in this solution formulation and they can be identified in the expressions for δI_1, δI_2, δI_3 in Eqs. (5-11). Let:

$$\left.\begin{aligned}
\delta I_1 &= \sum_{m=1}^{K} \sum_{n=1}^{L} \frac{TK^3}{L}\, \delta \underset{\sim}{Y}^{*T}_{(m,n)}\, \underset{\sim}{B}\, \underset{\sim}{Y}_{(m,n)} \\
\delta I_2 &= \sum_{m=1}^{K} \sum_{n=1}^{L} (P-g\sin\alpha)\, \frac{T}{L}\, \delta \underset{\sim}{Y}^{*T}_{(m,n)} \left[(K-m+1)\underset{\sim}{A}-\underset{\sim}{D}\right]\underset{\sim}{Y}_{(m,n)} \\
\delta I_3 &= \sum_{m=1}^{K} \sum_{n=1}^{L} \left(-\frac{L}{TK}\right) \delta \underset{\sim}{Y}^{*T}_{(m,n)}\, \underset{\sim}{C}\, \underset{\sim}{Y}_{(m,n)}
\end{aligned}\right\} \qquad (5\text{-}13)$$

Then

$$\left.\begin{aligned}
\underset{\sim}{A} &= \int_0^1 \int_0^1 \underset{\sim}{a}_{,\xi}\underset{\sim}{a}^T_{,\xi}\, d\xi d\eta \\
\underset{\sim}{B} &= \int_0^1 \int_0^1 \underset{\sim}{a}_{,\xi\xi}\underset{\sim}{a}^T_{,\xi\xi}\, d\xi d\eta \\
\underset{\sim}{C} &= \int_0^1 \int_0^1 \underset{\sim}{a}_{,\eta}\underset{\sim}{a}^T_{,\eta}\, d\xi d\eta \\
\underset{\sim}{D} &= \int_0^1 \int_0^1 \xi\, \underset{\sim}{a}_{,\xi}\underset{\sim}{a}^T_{,\xi}\, d\xi d\eta
\end{aligned}\right\} \qquad (5\text{-}14)$$

Each of these matrices $\underset{\sim}{A}$, $\underset{\sim}{B}$, $\underset{\sim}{C}$ or $\underset{\sim}{D}$ is of size 16 x 16 and thus has 256 values to be evaluated. Since they are independent of the element parameters, they need only to be computed once for all the elements.

5.3 Element Matrices Due to Discontinuous Load Function. These element matrices are identified with the terms of δI_4 through δI_{11} in Eqs. (5-11) where a discontinuous function $H_{(m,n)}(\bar{\xi}-\xi)$, $\bar{\delta}_{(m,n)}(\bar{\xi}-\xi)$, or their derivative is involved in the integration. It is clear that the line of discontinuity is defined by $\xi = \bar{\xi}(\eta)$. The specific form of $\bar{\xi}(\eta)$ is now given. Since we assume that the projectile moves with a constant acceleration β and that the velocity and position are zero at $t = 0$, the projectile position can be written as

$$x = \bar{x} = \frac{1}{2}\beta t^2 \tag{5-15}$$

From Eqs. (5-4), one has

$$\xi = \bar{\xi} = -m+1+\frac{\beta T^2 K}{2L^2}(\eta+n-1)^2 \tag{5-16}$$

Eq. (5-16) is obviously dependent on the element parameter m,n. In addition, the integrations of δI_4 through δI_{11}, will depend on whether and how the curve of Eq. (5-15) actually goes through the particular element (m,n). Thus these element matrices must be evaluated element by element.

5.4 Boundary Element Matrices and Non-Self-Adjoint Matrices. The contribution due to the end elements are identified with the δI_{12} and δI_{13} terms. Let

$$\delta I_{12} = \sum_{n=1}^{L} \left(\frac{T}{L}\right) [\delta \underset{\sim}{Y}^{*T}_{(1,n)} [k_1 \underset{\sim}{B}_1 + k_2 K^2 \underset{\sim}{B}_2] \underset{\sim}{Y}_{(1,n)}$$

$$+ \delta \underset{\sim}{Y}^{*T}_{(K,n)} [k_3 \underset{\sim}{B}_3 + k_4 K^2 \underset{\sim}{B}_4] \underset{\sim}{Y}_{(K,n)}]$$

$$\delta I_{13} = \sum_{m=1}^{K} \delta \underset{\sim}{Y}^{*T}_{(m,L)} \frac{k_7}{L} \underset{\sim}{B}_7 \underset{\sim}{Y}_{(m,1)}$$

Then, it is clear from Eqs. (5-12) that

$$\underset{\sim}{B}_1 = \int_0^1 \underset{\sim}{a}(0,\eta) \underset{\sim}{a}^T(0,\eta) d\eta$$

$$\underset{\sim}{B}_2 = \int_0^1 \underset{\sim}{a}_{,\xi}(1,\eta) \underset{\sim}{a}^T_{,\xi}(1,\eta) d\eta$$

$$\underset{\sim}{B}_3 = \int_0^1 \underset{\sim}{a}(1,\eta)\underset{\sim}{a}^T(1,\eta)d\eta$$

$$\underset{\sim}{B}_4 = \int_0^1 \underset{\sim}{a}_{,\xi}(1,\eta)\underset{\sim}{a}^T_{,\xi}(1,\eta)d\eta$$

and

$$\underset{\sim}{B}_7 = \int_0^1 \underset{\sim}{a}(\xi,1)\underset{\sim}{a}^T(\xi,0)d\xi$$

It should be mentioned that I_{13} is the only term in the present problem which contributed to the non-self-adjointness of the problem. This is also manifested in the element matrix $\underset{\sim}{B}_7$ which is the only nonsymmetric matrix.

5.5 Element Force Vectors. These force vectors are defined in conjunction with δJ_1, δJ_2 and δJ_3 in Eqs. (5-12). Let

$$\delta J_1 = \sum_{m=1}^{K}\sum_{n=1}^{L}\left(-\frac{Tg\cos\alpha}{KL}\right)\delta \underset{\sim}{Y}^{*T}_{(m,n)}\underset{\sim}{F}_1$$

$$\delta J_2 = \sum_{m=1}^{K}\sum_{n=1}^{L}\left(-\frac{Tgm\cos\alpha}{L}\right)\delta \underset{\sim}{Y}^{*T}_{(m,n)}\underset{\sim}{F}_2$$

$$\delta J_3 = \sum_{m=1}^{K}\left(\frac{k_7}{K}\right)\delta \underset{\sim}{Y}^{*T}_{(m,L)}\underset{\sim}{F}_3$$

Then

$$\underset{\sim}{F}_1 = \int_0^1\int_0^1 \underset{\sim}{a}(\xi,\eta)d\xi d\eta$$

$$\underset{\sim}{F}_2 = \int_0^1\int_0^1 a(\xi,\eta)\bar{\delta}_{(m,n)}(\bar{\xi}-\xi)d\xi d\eta$$

$$F_3 = \int_0^1 Y_{(m)}(\xi)\underset{\sim}{a}(\xi,1)d\xi$$

5.6 Flow Chart of the Computer Program. As shown in Figure 5 the computations in the present program consist of two parts: data preparation and solution program.

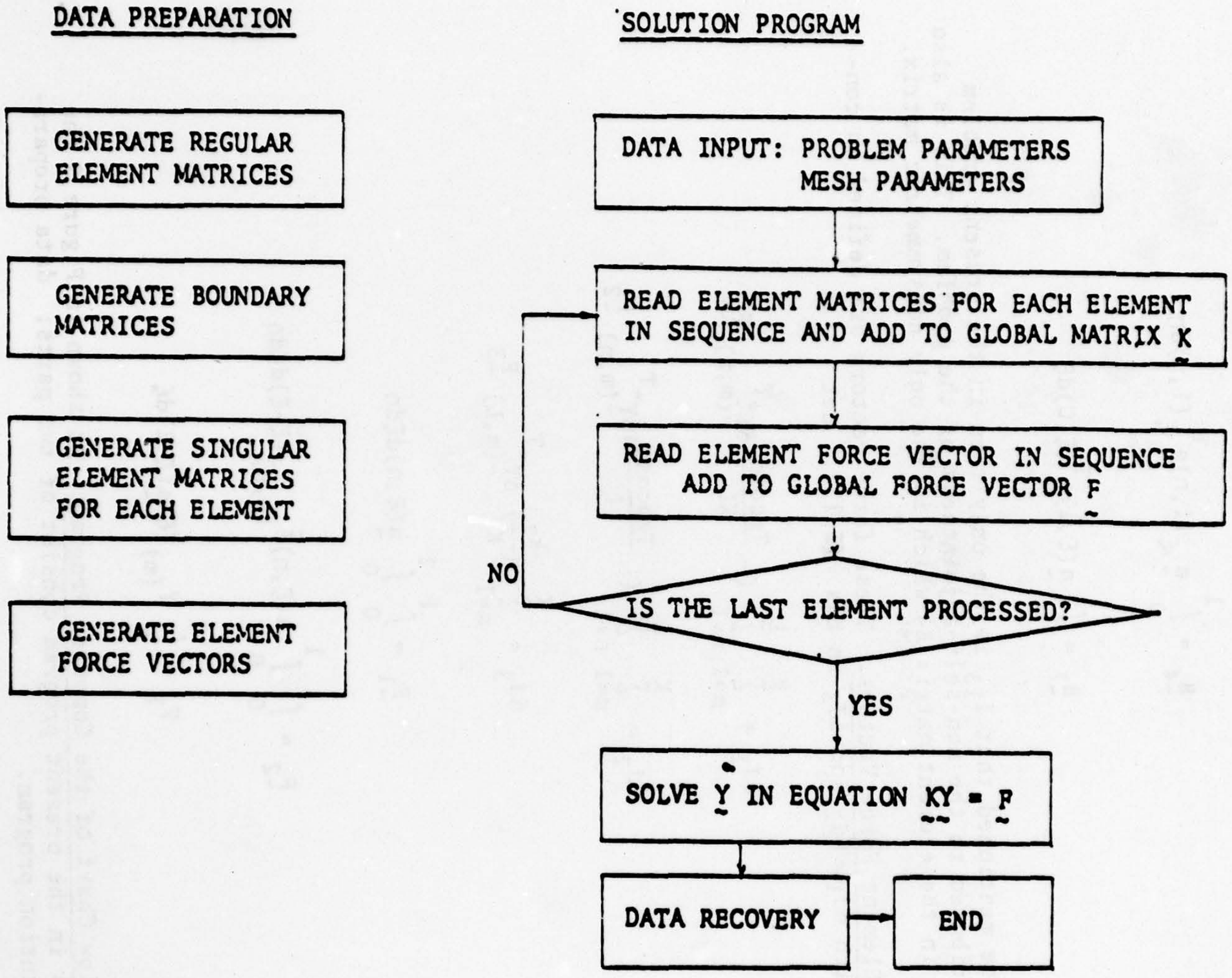

DATA PREPARATION
GENERATE REGULAR ELEMENT MATRICES
GENERATE BOUNDARY MATRICES
GENERATE SINGULAR ELEMENT MATRICES FOR EACH ELEMENT
GENERATE ELEMENT FORCE VECTORS
SOLUTION PROGRAM
DATA INPUT: PROBLEM PARAMETERS MESH PARAMETERS
READ ELEMENT MATRICES FOR EACH ELEMENT IN SEQUENCE AND ADD TO GLOBAL MATRIX $\underset{\sim}{K}$
READ ELEMENT FORCE VECTOR IN SEQUENCE ADD TO GLOBAL FORCE VECTOR $\underset{\sim}{F}$
IS THE LAST ELEMENT PROCESSED?
NO
YES
SOLVE $\underset{\sim}{Y}$ IN EQUATION $\underset{\sim}{K}\underset{\sim}{Y} = \underset{\sim}{F}$
DATA RECOVERY
END

FIGURE 5. Flow Chart of the Computer Program.

6. DEMONSTRATIVE RESULTS OF COMPUTATIONS. Some preliminary results have been obtained by the computer program described in previous sections. Our present objective is simply to demonstrate that the concept of the unconstrained variational formulation works for the initial boundary value problem of gun dynamics, that the initial data can indeed be recovered by the unconstrained process and that the nonconservative and highly discontinuous nature of the loads can be tested in a routine and rather elegant manner. Thus the numerical values obtained are not considered final. Their verifications will be carefully performed and presented in a report to follow.

The numerical values for those nondimensional parameters listed in Section 2 are given here.

$$T = 0.07695$$

$$\bar{P} = 4.8025$$

$$m = 0.012$$

$$\beta = 308.79$$

$$\alpha = 0.5236$$

$$g = 0.020$$

These data are calculated from an approximate model for a 105 mm M68 cannon with the following material and dimensional parameters:

E (Young's modulus) = $30\text{x}10^6$ psi

ρ (density) = 0.2818 lb/in^3

I.D. (of gun tube) = 105 mm = 4.13"

O.D. = 7.28" (average of 5.69" muzzle
O.D. and 8.88", breech end O.D.)

$A = 28.23\ \text{in}^2$

$I = 123.60\ \text{in}^4$

$\ell = 17.5' = 210''$

$$c = \left(\frac{\rho A \ell^4}{EI}\right)^{1/2} = 0.10396\ \text{sec.}$$

m_p (projectile mass) = 210 lb

P (bore pressure) = 30,000 psi

β_p (projectile acceleration) = 500,000 ft/sec^2

The initial position of the tube is assumed to be the static deflection of the tube under gravitational force $\bar{q} = \rho A g$. Hence

$$Y(x) = \frac{q}{24}(x^4-4x^3+6x^2)$$

where $q = \frac{c^2 g}{\ell}$ and g is the gravitational acceleration. Hence q = 0.01987.

Using the data given above, the solution of a test problem is shown in Tables III and IV. The exact values of the initial data, which corresponds to the static deflection of the tube, are given in the parentheses. From these tables, it is clearly shown that the given initial data have indeed been recovered in the solutions.

TABLE III. CALCULATED $y(x,t)(x10^{-3})$ FROM THE PRESENT COMPUTER PROGRAM (EXACT INITIAL DATA IN PARENTHESES)

t \ x	0	0.2	0.4	0.6	0.8	0.10
0	0.00000 (0.00000)	0.17477 (0.17467)	0.60803 (0.60800)	1.18801 (1.18800)	1.83469 (1.83467)	2.49996 (2.50000)
0.5T	0.00000	0.19504	0.59019	1.17861	1.82291	2.48327
T	0.00000	0.21705	0.58048	1.15876	1.77023	2.45612

TABLE IV. CALCULATED $\frac{\partial y(x,t)}{\partial x}$ $(x10^{-3})$ FROM THE PRESENT COMPUTER PROGRAM

(EXACT INITIAL DATA IN PARENTHESES)

t \ x	0	0.2	0.4	0.6	0.8	0.10
0	0.00000 (0.00000)	1.62731 (1.62667)	2.61662 (2.61333)	3.12140 (3.12000)	3.30629 (3.30667)	3.32996 (3.33333)
0.5T	0.00000	1.63528	2.62714	3.14336	3.27633	3.31506
T	0.00000	2.06887	1.68236	3.03449	3.23987	3.55142

REFERENCES

1. J. J. Wu, "Solution to Initial Value Problems by Use of Finite Element-Unconstrained Variational Formulation," Journal of Sound and Vibration (1977), Vol. 53, pp. 341-356.

2. J. J. Wu, "Column Instability Under Nonconservative Forces, With Internal and External Damping - Finite Element Using Adjoint Variational Principles," Developments in Mechanics (1973), Vol. 7, pp. 501-514.

3. J. J. Wu, "A Unified Finite Element Approach to Column Stability Problems," Developments in Mechanics (1975), Vol. 8, pp. 279-294.

4. J. J. Wu, "Gun Dynamics Analysis by the Use of Unconstrained, Adjoint Variational Formulations," Proceedings of the Second U.S. Army Symposium on Gun Dynamics, Watervliet, NY, September 1978, pp. II 81-II 99.

5. T. E. Simkins, "Radial and Transverse Response of Gun Tubes by Finite Element Methods," Proceedings of the First Conference on Dynamics of Precision Gun Weapons, Rock Island, IL, January 1977, pp. 373-469.

6. L. Frýba, Vibration of Solids and Structures Under Moving Loads, Noordhoff International Publishing Groningen, (1971), p. 317.

A PRELIMINARY INVESTIGATION OF THE SINGULAR BEHAVIOR OF FLUIDS NEAR A SLIDING CORNER

Patrick J. Roache*
Csaba K. Zoltani
Ballistic Research Laboratory
Aberdeen Proving Ground, MD 21005

ABSTRACT. This paper describes the application of a semidirect method to the calculation of the flow near a sliding corner. On a set of successively refined grids the Navier-Stokes equations modeling the steady state viscous, incompressible, low Reynolds number flow are solved in terms of the vorticity and stream function in Cartesian coordinates for a slab geometry. The calculated flow field conforms to theoretical expectations and allows some preliminary estimate of the nature of the physical boundary condition at the sliding corner.

1. INTRODUCTION. One of the vexing problems in the implementation of multidimensional computer codes for interior ballistics applications is the proper handling of the flow singularity at the conjunction of the gun tube wall and the base of a moving projectile. The value of an incorrectly determined flow variable at this point is likely to be propagated into the interior of the flow, possibly leading to disastrous consequences for the rest of the flow field. In the past, the corner point singularity was treated by ignoring it, i.e., by ensuring that the computational stencil does not fall on the corner point, or by assuring that the corner is multivalued, i.e., it is both at rest and moving with the piston velocity. Neither of these approaches is completely satisfactory. The nature of the flow immediately adjacent to the corner should suggest the correct boundary condition at the corner point itself. Consequently, we apply a series of grid refinements and examine the nature of and changes in the flow field as we approach the sliding corner point.

2. THE MODEL PROBLEM. Consider a smooth bore gun tube at an early time in the ballistic cycle. The projectile, propelled by the high pressure combustion gases proceeds down the gun tube. For the case at hand, the obturation is perfect, blowby is not allowed. The flow field behind the accelerating projectile consists of a core flow, a boundary layer on the gun tube walls, and the corner flow.

Looking at a slab cut out of this tube along any diameter, we have the following configuration: a rectangular region bounded on the right by a moving piston, below by a smooth adiabatic impermeable wall, above by the axis of symmetry and at the left by the inflowing gas with a given velocity distribution.

* Ecodynamics Research Associates, Inc., Albuquerque, NM 87108. Sponsored under Contract DAAG29-76-C-0100.

The flow is assumed to be incompressible, viscid, steady and of constant density throughout the region of interest. From the Navier-Stokes equations, and the definition of vorticity

$$\zeta = \frac{\partial u}{\partial y} - \frac{\partial v}{\partial x} \qquad (1)$$

one obtains the vorticity transport equation

$$\frac{\partial \zeta}{\partial t} = -u\,\frac{\partial \zeta}{\partial x} - v\,\frac{\partial \zeta}{\partial y} + \nu\left(\frac{\partial^2 \zeta}{\partial x^2} + \frac{\partial^2 \zeta}{\partial y^2}\right) \qquad (2)$$

With the definition of the stream function

$$\frac{\partial \psi}{\partial y} = u,\ \frac{\partial \psi}{\partial x} = -v \qquad (3)$$

equation (1) transforms into the Poisson equation

$$\nabla^2 \psi = \zeta \qquad (4)$$

Introducting a characteristic length, L, the tube radius, and a time defined as the advective time constant ($L/\bar{u}_0$), with $\bar{u}_0$ a characteristic velocity, the equation may be brought into the following nondimensional form

$$\zeta_t = -\text{Re}\ \nabla \cdot \underline{v}\ \zeta + \nabla^2 \zeta \qquad (5)$$

and

$$\nabla^2 \psi = \zeta \qquad (6)$$

where ζ is the vorticity nondimensionalized by ($\bar{u}_0/L$), Re is the Reynolds number, $\underline{v}$ the vector velocity. Subscripts denote partial differentiation. Since we are looking at the steady case only, $\zeta_t = 0$. A moving coordinate system is now introduced which brings the projectile to rest. One of the advantages which accrues is that one does not have to deal with partial cells in the course of the calculation.

3. <u>THE BOUNDARY CONDITIONS</u>. Along the inflow boundary, Poiseuille flow, and thus the value of the stream function is specified. The upper boundary is the axis of the tube, a line of symmetry at which $\zeta = 0$.

In the transformed coordinate system the lower boundary moves to the left at a velocity of $u = -1$ and ψ, the stream function is set equal to zero. ζ_w, the wall vorticity, is calculated from Woods' equation; see reference 1 for details.

The basic idea is to expand the stream function in a Taylor series around a point one mesh width away from the wall retaining terms up to the third order. Using the definition of vorticity and applying the no slip condition at the face of the piston (the right hand termination of the computational grid) one arrives at

$$\zeta_w = \frac{3(\psi_{w+1} - \psi_w)}{\Delta y^2} - \tfrac{1}{2}\zeta_{w+1}$$

where the subscripts w+1 and w denote a point one mesh width away from the wall and the wall point, respectively. A similar expression holds at the lower wall.

4. THE ALGORITHM. A semidirect solution technique was used, motivated by the efficiency of these methods. As in [2], the equations are first linearized, then solved directly, and then the linearization is updated and the iteration is repeated. Unlike the method in [1], the variable coefficient advection terms are included in the direct linear solution by the use of marching methods [3].

The marching method is unstable for elliptic partial differential equations as the mesh is refined, i.e., as $\Delta \ell \to 0$ where $\Delta \ell$ is the mesh size. For finite mesh widths, however, usable results may be obtained.

Converged answers are typically obtained in less than 10 iterations. The total computational time on a CDC 6600 is of the order of 5-10 msec/cell. No empirical or semi-empirical factors (such as time steps, under-relaxation factors, etc.) need to be determined.

5. THE CALCULATIONAL SET UP. A series of calculations were performed on meshes varying from 11x11 to 31x31 and in the Reynolds number range from 0 (to initialize) to 50. The cell aspect ratio was varied from 1:1 to 4:1. The latter was required by reasons of stability.

To start the calculation, the upstream flow profile was prescribed (Poiseuille flow) and the lower wall was given an instantaneous velocity of -1 to the left. Reynolds number was set at zero. Then, in increments of 10, the Reynolds number was raised up to 50.

Several checks monitored the progress of the calculation. If the calculation failed to converge in 21 iterations, the calculation was broken off, and an error message printed. In addition, the results of two convergence checks were printed out.

6. DESCRIPTION OF THE RESULTS. The results are best described in terms of plots of the flow variables. The first figure shows the velocity profile at the position 0.6 from the upstream (inflow) boundary for Re = 30. It is the Poiseuille profile in the transformed coordinates.

Next the contour plot of the u velocity component gives an idea of the magnitudes of the flow velocity in the computational regime. An even better overall view can be obtained from the vector velocity plot, Figure 3. Note

AD-A074 800   ARMY RESEARCH OFFICE RESEARCH TRIANGLE PARK NC   F/G 5/9

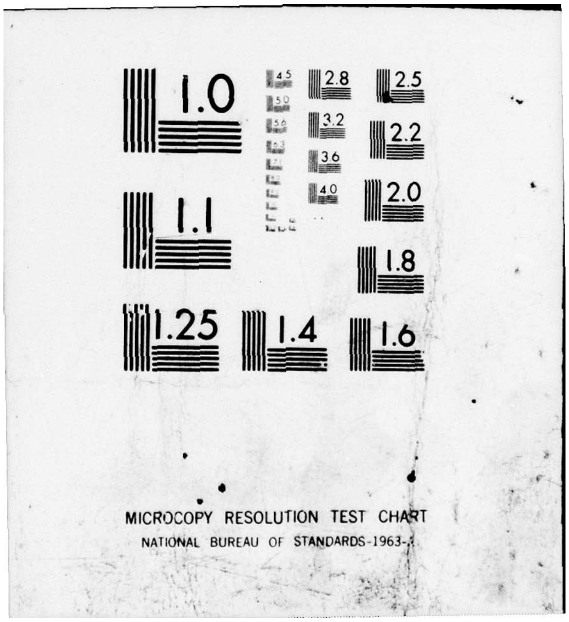
PROCEEDINGS OF THE 1979 ARMY NUMERICAL ANALYSIS AND COMPUTERS C--ETC(U)

SEP 79

UNCLASSIFIED   ARO-79-3   NL

4 OF 6

AD A074 800



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

that the flow (in the transformed coordinates) is parallel to the tube axis, then along the face of the piston, finally turning around and exiting. By incompressibility the flow in and out of the region of interest balance exactly.

The best insight into the nature of the flow singularity in the corner, of course, is furnished by the vorticity plot, Figures 4-7. While the contour plot gives a good overall idea of the vorticity distribution, more detail is furnished by vorticity plotted against axial distance along the wall as well as vorticity along the piston face.

In both instances we see that near the corner singularity the vorticity rises sharply. This is readily seen in the three dimensional plot, Figure 7.

Significantly, we found that the vorticity is increased only slightly as the Reynolds number was raised from 10 to 50. Mesh refinement has a pronounced effect on the maximum level of vorticity, being lower for the coarser mesh, Figure 8. This is expected since the fine mesh puts points closer to the singularity. One should note, however, that these quantitative results must be interpreted carefully, since in these preliminary runs the length to width ratio of the cells was not kept constant.

7. CONCLUSIONS AND OUTLOOK. At this point our conclusions are preliminary in nature. We found, through mesh refinement near the confluence of the piston and a moving wall that the calculated flow there conforms to qualitative estimates.

As one proceeds radially from the corner, the velocity is seen to increase. This is true along all radii in the sector formed by the piston and the cylinder wall.

The vorticity function exhibits a sharp rise near the corner and one is reminded of a sink type flow behavior.

We expect to continue our numerical experimentation and anticipate that analogous flow behavior will be found for other mesh ratios as well as for calculation in cylindrical coordinates. The cylindrical coordinate calculations are now in progress, using exponential coordinate transformations to increase resolution near the corner.

REFERENCES

1. Roache, P. J., Computational Fluid Dynamics, Revised Printing, Hermosa Publishers, Albuquerque, New Mexico, 1976.

2. Roache, P. J., and Ellis, M. A., "The BID Method for the Steady-State Navier-Stokes Equations", Computers and Fluids, 3, 305-320 (1975).

3. Roache, P. J., "Marching Methods for Elliptic Problems: Part 1," Numerical Heat Transfer 1, 1-25 (1978). "Part 2", 1, 163-181 (1978).

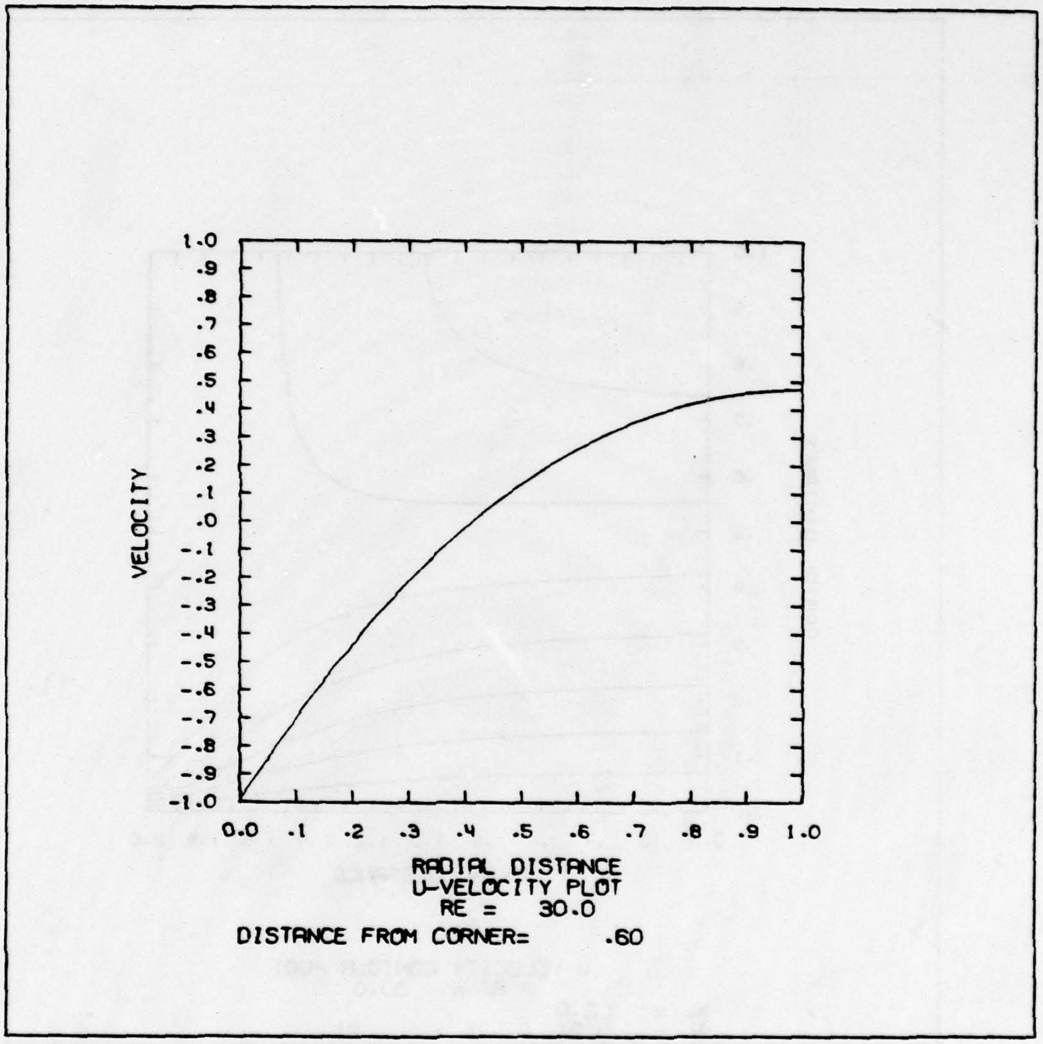


Figure 1. Velocity Profile at a Position 0.6 from the Inflow Boundary at Re=30.

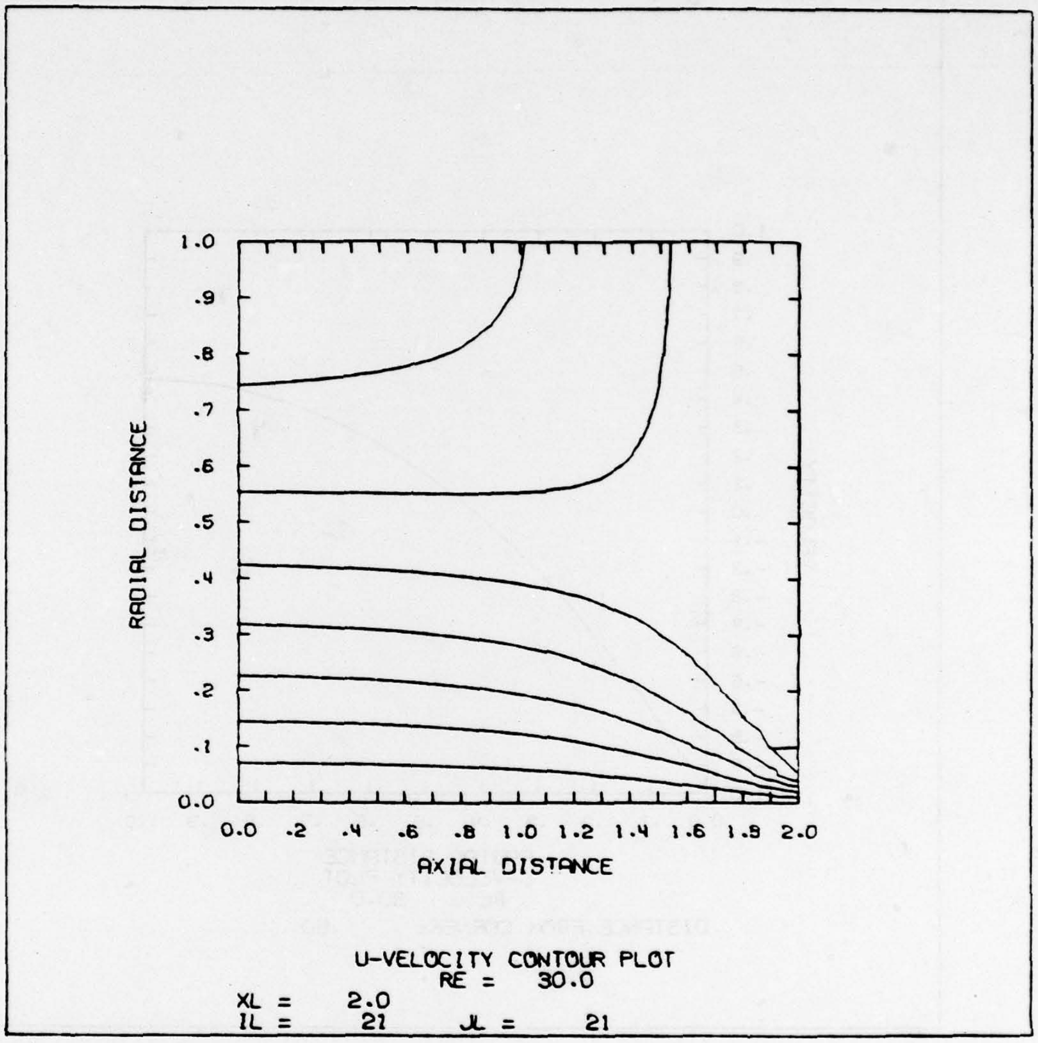


Figure 2. Contour Plots of the Axial Velocity Component in the Computational Region.

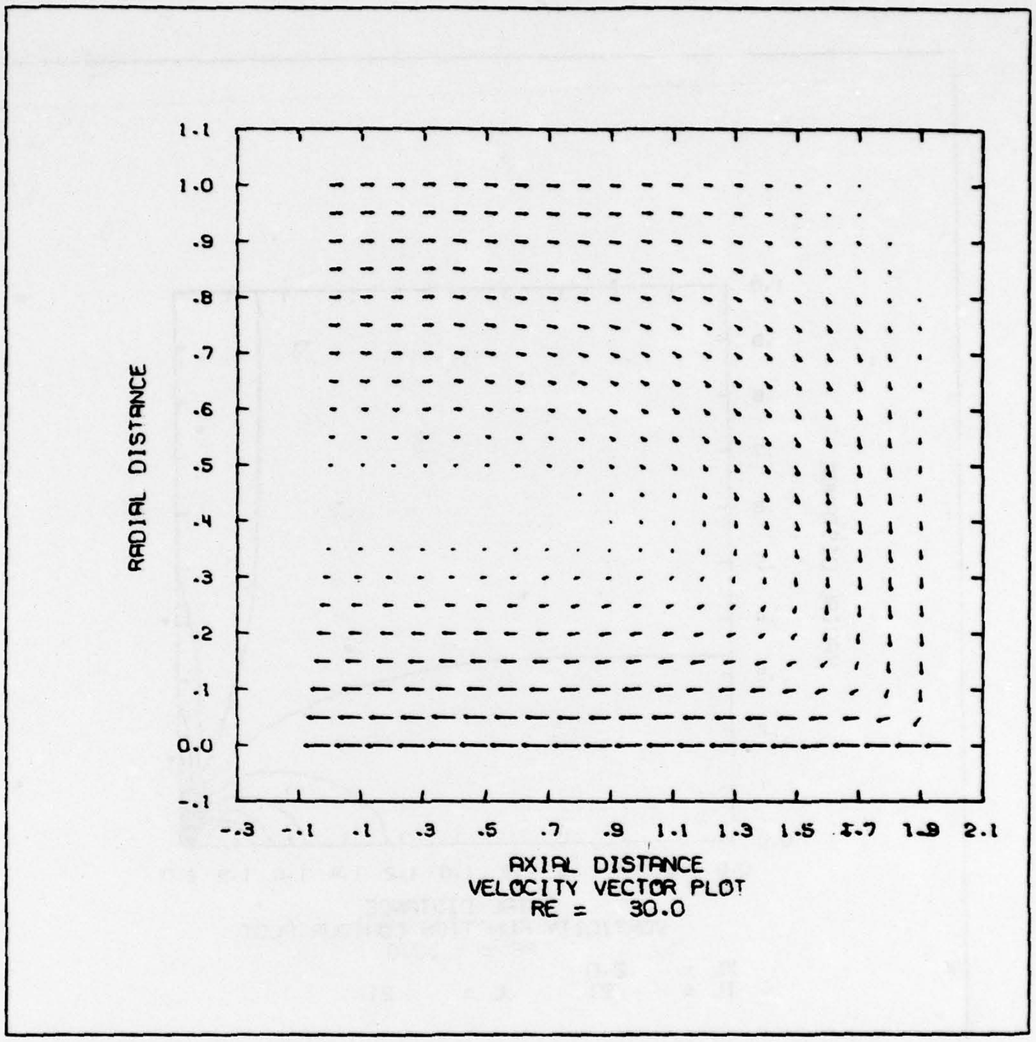


Figure 3. Vector Velocity Plot.

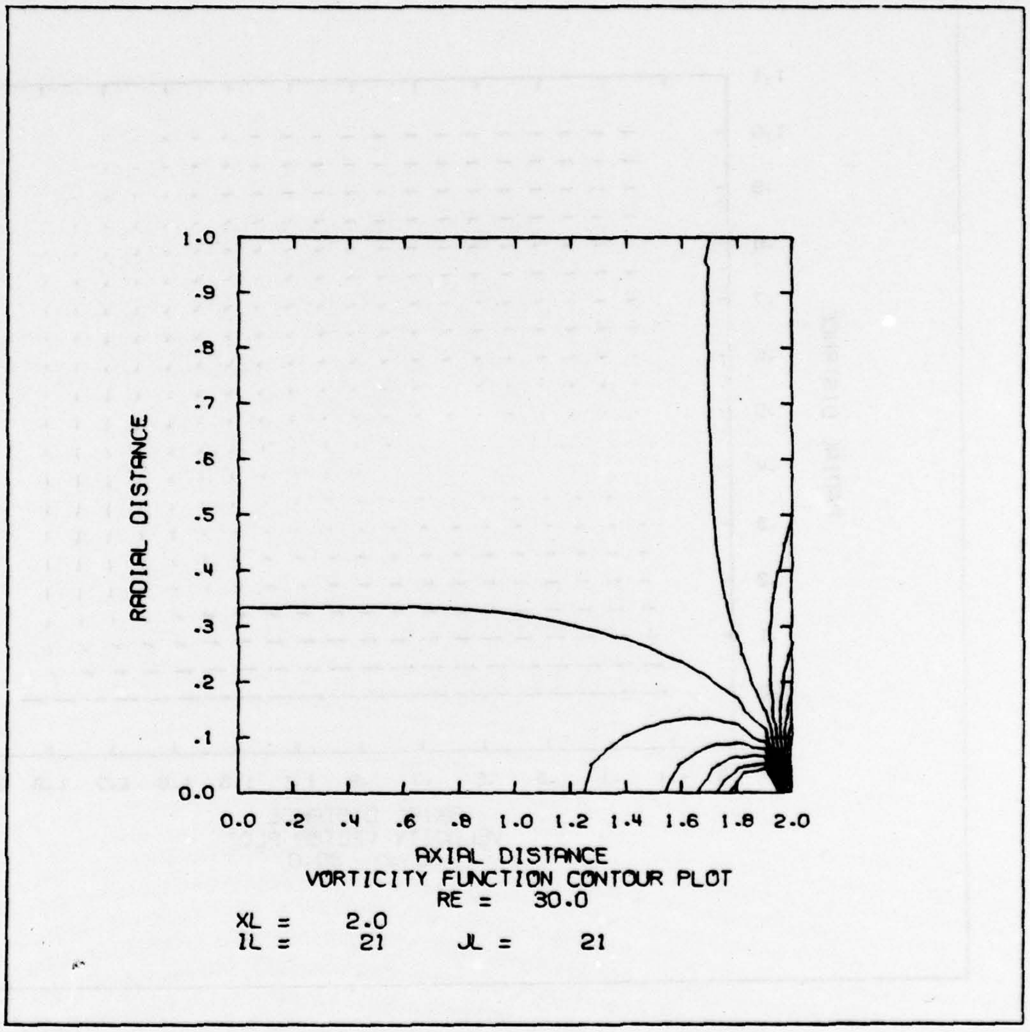


Figure 4. Vorticity Function Contour Plot at Re=30.

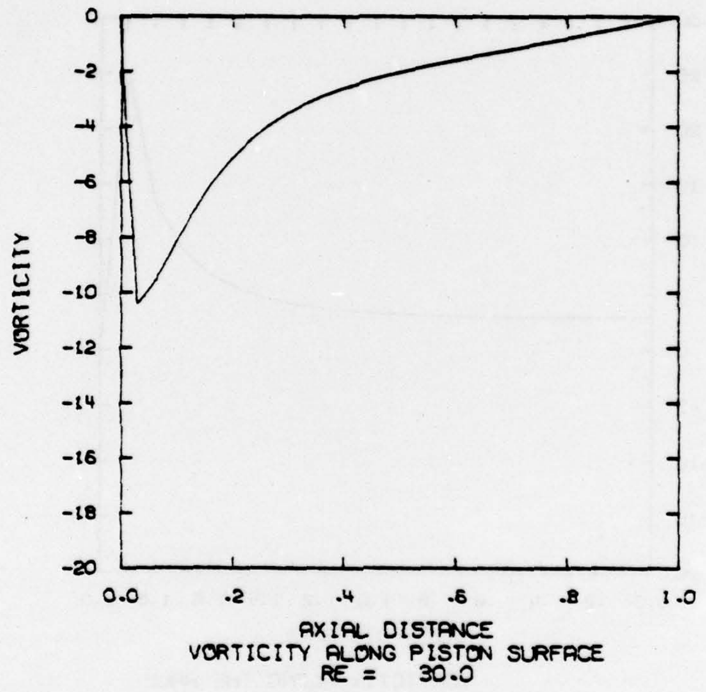


Figure 5. Vorticity Along the Piston Surface.

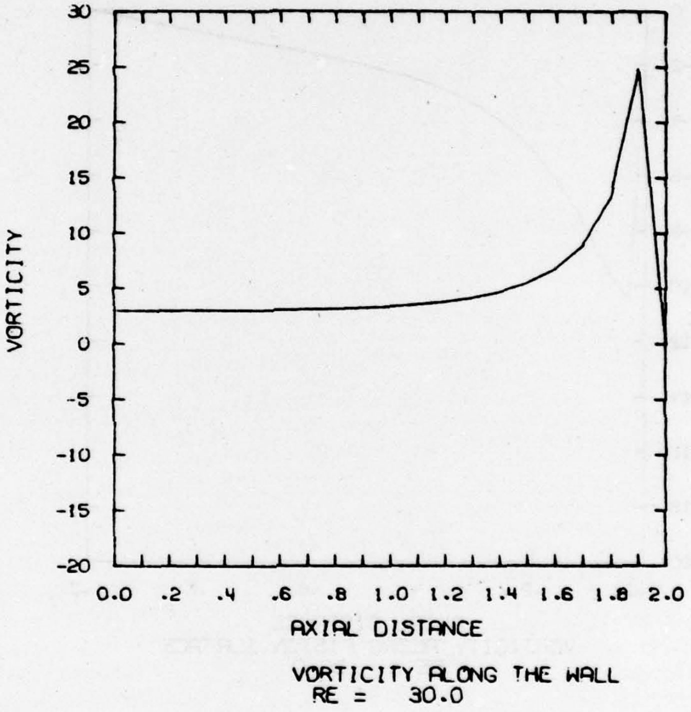


Figure 6. Vorticity Along the Lower Wall.

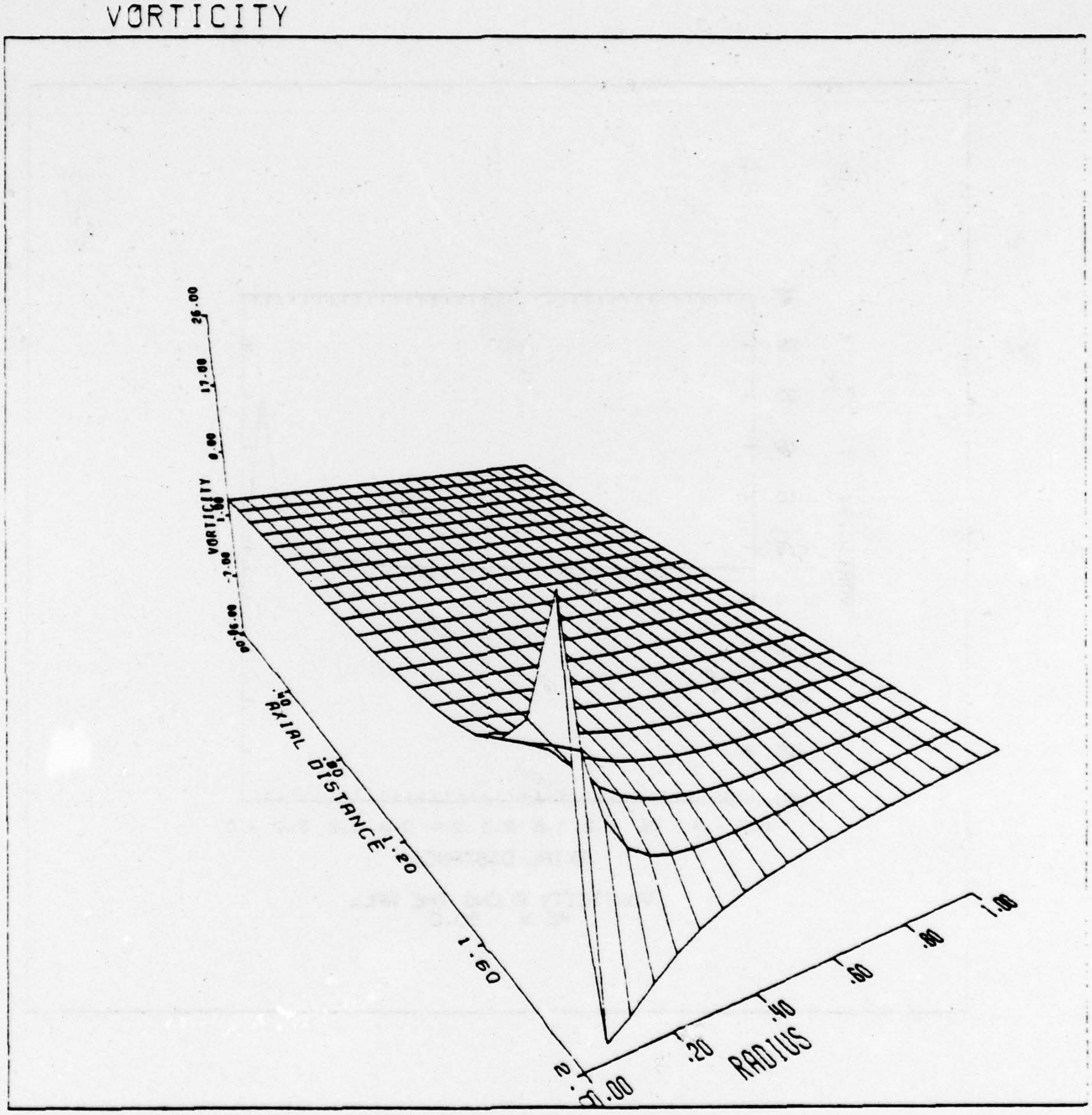


Figure 7. Vorticity Surface Plot in the Computational Domain.

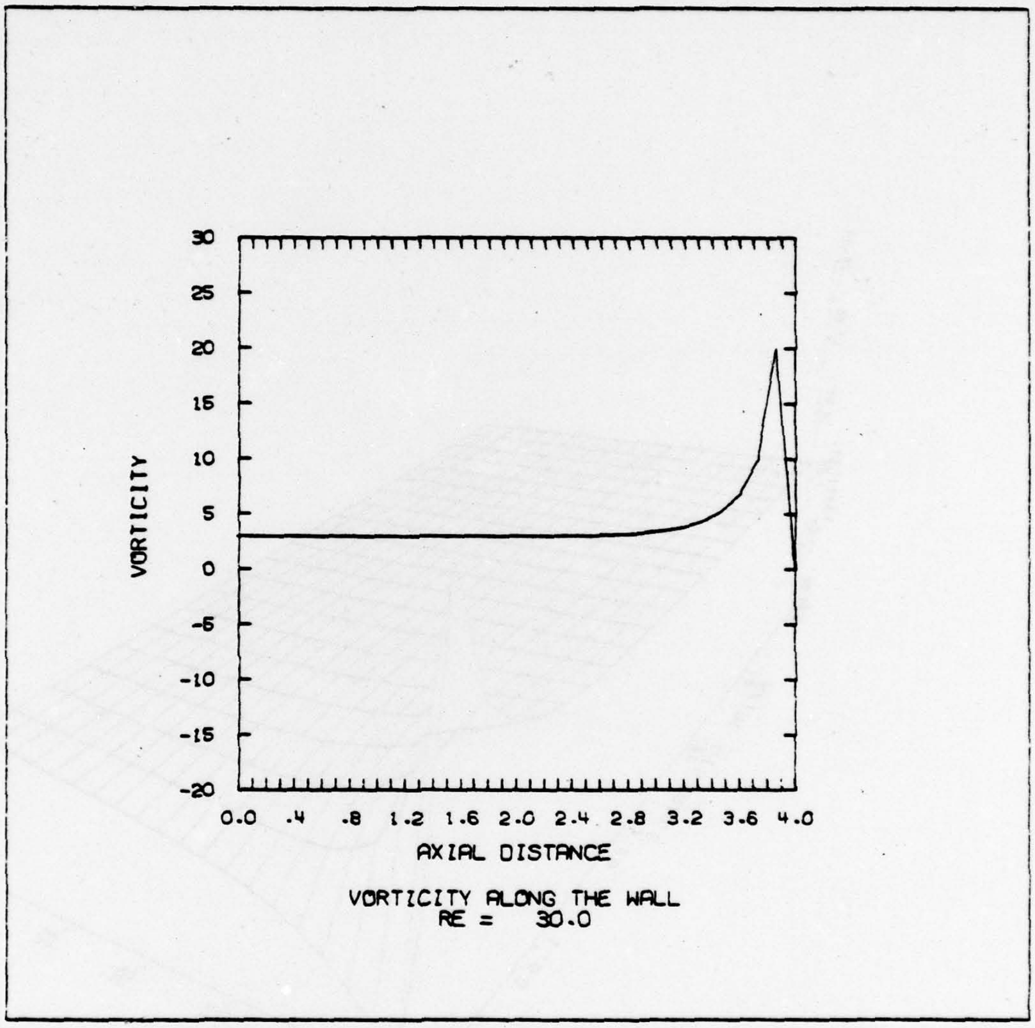


Figure 8. Vorticity Along the Wall for Coarser Mesh Definition.

NUMERICAL SOLUTION OF THE SAD METHOD FOR ROBUST REGRESSION

William S. Agee
and
Robert H. Turner
Mathematical Services Branch
Analysis and Computation Division
National Range Operations Directorate
White Sands Missile Range, New Mexico 88002

ABSTRACT.

Given observations $y_i = X_i\theta + e_i$, where θ is to be estimated, it is well known that estimating θ by L_1 minimization is much less sensitive to outliers among the observation than estimation by least squares. A method related to L_1 minimization which is also robust is minimizing the Sum of Absolute Deviations (SAD). The SAD method minimizes $\sum_{i=1}^{n} |r_i - r^*|$ where $r_i = y_i - X_i\theta$ are the residuals and r^* is the median of these residuals $r^* = \underset{i=1,n}{\text{median}}\ r_i$. An iterative method for the numerical solution of the SAD regression is given along with examples of its use in the presence of multiple outliers. The iterative method used for solution does not seem to be computationally efficient and has some problems with convergence. A more computationally efficient and stable method for solution might be developed by modifying one of the finite step methods available for L_1 minimization.

1. INTRODUCTION.

The presence of outliers in a set of observations makes the reliable estimation of parameters in a linear model much more difficult than when no outliers are present. The usual methods for estimation of the parameters such as least squares and maximum likelihood produce nearly useless results when outliers are present in the observations. Methods for robust regression offer a fresh approach to the problem of parameter estimation in the presence of outliers. In data reduction applications, this problem is usually called the data editing or observation editing problem. Robust regression has proven highly successful in treating this problem in several data reduction applications [1] and [2].

Robust estimation methods have been classified by Huber [3] and [4]. Huber's classifications are termed L-estimates, M-estimates, and R-estimates. The L-estimates are formed as linear combinations of order statistics. The M-estimates for which Huber has been the leading proponent are similar to maximum likelihood estimates in that they are formed by minimizing a pseudo likelihood function $\prod_i e^{-\rho_i}$ where $e^{-\rho_i}$ is a pseudo density of the residuals with ρ_i a suitable function of the residuals. The R-estimates are derived on the basis of rank tests. The applications of robust regression to data reduction problems described in [1] and [2] were based on M-estimates.

Given observations $y_i = X_i^T \theta + e_i$, $i = 1, N$ where θ is a parameter vector to be estimated, it is well known that estimating θ by L_1 minimization is much less sensitive to outliers among the observations than estimation by least squares. A method for estimating θ, related to L_1 minimization, which is even more robust in the presence of outliers is minimizing the Sum of Absolute Deviations (SAD). The SAD method minimizes $\sum_{i=1}^{n} |r_i - r^*|$ where $r_i = y_i - X_i \theta$ and r^* is the median of those residuals, $r^* = \operatorname{med}_{i=1,N} r_i$. The SAD method for robust regression is an R-estimate. The following describes some properties of the SAD estimate, some methods for numerical computation of SAD estimates, and the application of the SAD method to some real data having multiple outliers.

2. THE SAD METHOD OF REGRESSION.

Given the linear model

$$(1) \qquad y_i = X_i^T \theta + e_i, \; i=1,N \quad ,$$

where $X_i^T = [X_{i_1}, X_{i_2} \dots X_{im}]$ and θ is a m-vector to be estimated. Let r_i be the residuals

$$(2) \qquad r_i = y_i - X_i^T \theta \; .$$

The SAD method minimizes

$$(3) \qquad \sum_{i=1}^{n} |r_i - r^*| \; ,$$

where r^* is the median of the residuals.

Hettmansperger and McKean [5] describe a class of R-estimates which minimizes

$$(4) \qquad \sum_{i=1}^{n} a(R_i) r_i \; ,$$

where R_i is the rank of the residual r_i and $a(R_i)$ is a score function such that $\sum_{i=1}^{n} a(i) = 0$, $a(1) \leq \dots \leq a(N)$, and $a(i) = -a(N-i+1)$. If in (4) we use the median scores

(5) $$a(i) = \begin{cases} 1 & i > \frac{N+1}{2} \\ 0 & i = \frac{N+1}{2} \\ -1 & i < \frac{N+1}{2} \end{cases}$$

it is easy to see that (3) and (4) are identical. Thus the SAD regression is equivalent to the R-estimate of Hettmansperger and McKean with median score functions. Jaekel [6] shows that dispersion measures of the form (4) are non-negative and are continuous and convex functions of the regression coefficients. Due to the equivalence of SAD with the R-estimate, (3) is convex so that its minimization leads to a unique solution regardless of the starting point. This is a definite advantage of the SAD regression method over many of the M-estimates (with a redescending ψ-function a M-estimate is obtained via a non-convex minimization and will sometimes converge to the wrong solution).

3. NUMERICAL SOLUTION FOR THE SAD REGRESSION.

We have obtained the SAD regression estimates by using iterative methods to minimize (3). Although these iterative methods are simple to apply, they may be more time consuming than finite step methods for minimizing (3). The development of a finite step minimization method similar to the linear programming method or the method of Bartels, Conn, and Sinclair [7] for L_1 minimization would be desirable.

The method used for minimization of (3) is the iterative application of weighted least squares. This method was reported on by Schlossmacher [8] for L_1 minimization and has also been used successfully for obtaining M-estimates [1] and [8]. Let $\hat{\theta}^{(K)}$ be the estimate of θ at the $K^{\underline{th}}$ stage of minimizing (3). The residuals from this estimate are

(6) $$r_i^{(K)} = y_i - X_i^T\hat{\theta}^{(K)}$$

Let $r^{*(K)} = \underset{i=1,N}{\text{med}}\ r_i^{(K)}$. The estimate $\hat{\theta}^{(K+1)}$ is obtained by minimizing the weighted sum of squares

(7) $$\sum_{i\varepsilon I_K} w_i^{(K)} \left(r_i - r^{*(K)}\right)^2 ,$$

where

(8) $$w_i^{(K)} = \frac{1}{|r_i^{(K)} - r^{*(K)}|}$$

and I_K is the index set, $I_K = \left\{ i \,\middle|\, |r_i^{(K)} = r^{*(K)}| > \varepsilon \right\}$. Thus, at each state of iteration, we are excluding those residuals which are too near the median and would therefore have very large weights. Alternatively, we might limit the magnitude associated with a residual rather than excluding residuals from the minimization. The estimate $\hat{\theta}^{(K+1)}$ obtained by minimizing (7) is

$$(9) \qquad \hat{\theta}^{(K+1)} = M_K^{-1} \sum_{i \varepsilon I_K} W_i^{(K)} \left(X_i - X^{*(K)}\right)\left(y_i - y^{*(K)}\right)$$

$$(10) \qquad M_K = \sum_{i \varepsilon I_K} W_i^{(K)} \left(X_i - X^{*(K)}\right)\left(X_i - X^{*(K)}\right)^T$$

In (9) and (10) $X^{*(K)}$ is the value of independent variables associated with the median residual $r^{*(K)}$ and $y^{*(K)}$ is the observation associated with $r^{*(K)}$.

4. SOME EXAMPLES.

The first example considered is the Daniel and Wood data. This data has been used by several authors to illustrate robust regression methods. The data is taken from Daniel and Wood [9], Chapter 5, where it is examined in detail. The data is a sequence of 21 observations in three independent variables given below:

Observation No.	y	x_1	x_2	x_3	SAD Residuals
1	42	80	27	89	5.05
2	37	80	27	88	0
3	37	75	25	90	5.42
4	28	62	24	87	7.64
5	18	62	22	87	-1.21
6	18	62	23	87	-1.78
7	19	62	24	93	-1.00
8	20	62	24	93	0
9	15	58	23	87	-1.45
10	14	58	18	80	- .01
11	14	58	18	89	.51
12	13	58	17	88	.02
13	11	58	18	82	2.89
14	12	58	19	93	-1.81
15	8	50	18	89	1.17
16	7	50	18	86	0
17	8	50	19	72	- .39
18	8	50	19	79	.01
19	9	50	20	80	.50
20	15	56	20	82	1.62
21	15	70	20	91	-9.5

The regression coefficients estimated by the SAD method are θ_0 = 39.806, θ_1 = .8327, θ_2 = .5712, θ_3 = .0594. This iterative solution method converged quickly for this example. The residuals from the SAD method correctly exhibit observations 1, 3, 4, and 21 as outliers in agreement with the original researchers. The SAD method performed considerably better on this example than M-estimates which converge to the wrong solution on this example when the starting solution is not close to the final solution. The performance of the SAD method on this example created considerable enthusiasm for this method.

A second example is from some radar range data at WSMR. The following sequence of observations has time (t) as an independent variable. A quadratic model $\theta_0 + \theta_1 t + \theta_2 t^2$ is fit to the observations.

t	Range	SAD Residuals
1.4	632	- .003
1.45	632	66.26
2.00	61846	60506
2.05	122880	121490
2.20	1552	19.79
2.25	1589	11.86
2.30	1626	5.26
2.35	1663	.0003
2.40	1701	-2.93
2.45	1736	-7.53
2.50	1772	-9.79
2.55	1808	-10.73
2.60	1844	-10.33
2.65	1879	-9.59
2.70	1915	-6.53
2.75	1954	-2.13
2.80	1986	2.59
2.85	2021	8.66
2.90	2056	16.06
2.95	2091	24.79

This set of observation has also proved very difficult to correctly identify the outliers by other robust methods. The iterative method for SAD converged reasonably well for this example, although not as quickly as for the Daniel and Wood data. The outliers for this example are observations 1, 2, 3, and 4. The SAD method correctly identifies three of these outliers. Other robust methods also have difficulty in identifying the first observation as an outlier.

A final example is some azimuth measurements taken from a tracking camera at WSMR. Again, time is the independent variable and we fit a quadratic model to the observations. The following residuals are from SAD after three iterations.

Observation Number	y	SAD Residuals 3 Iterations
1	-1.70987	- .00472
2	-1.70942	- .00189

Observation Number	y	SAD Residuals 3 Iterations
3	-1.70893	.00029
4	-1.70845	.00179
5	-1.70793	.00264
6	-1.70841	.00281
7	-1.70682	.00238
8	-1.70626	.00123
9	-1.70571	- .00061
10	-1.70510	- .00307
11	-1.70450	- .00623
12	1.43777	3.13159
13	1.44602	3.13472
14	-1.70259	- .01966
15	1.44667	3.12309

After three iterations, the SAD method correctly identifies the outliers, but with more iteration the numerical method produces a solution which moves very slowly toward another solution.

The convergence properties of the weighted least squares iteration are slow and noisy. Thus, a different method of solution is desirable. Hopefully, one of the finite step methods, such as the recently published algoritnm of Bartels, Conn, and Sinclair [7], could be modified to apply to the SAD regression.

5. REFERENCES.

1. Agee, W.S., and Turner, R.H., "Application of Robust Statistical Methods to Data Reduction," Analysis and Computation Division, Tech. Rpt. No. 65, White Sands Missile Range, March 1978.

2. Agee, W.S., and Turner, R.H., "Robust Regression: Computational Methods for M-Estimates," Analysis and Computation Division, Tech. Rpt. No. 66, June 1978, White Sands Missile Range, June 1978.

3. Huber, Peter J., "Robust Regression: Asymptotics, Conjectures, and Monte Carlo," Annuals of Statistics, 1, (1973), p 799-821.

4. Huber, Peter J., "Robust Statistics; A Review," Annuals of Mathematical Statistics, 43, (1972), p 1041-1067.

5. Hettmansperger, T.H., and McKean, J. W., "A Robust Alternative Based on Ranks to Least Squares in Analyzing Linear Models," Technometrics, 19, (August 1977), p 275-284.

6. Jaekel, L. A., "Estimating Regression Coefficients by Minimizing the Dispersion of the Residuals," Annals of Math. Stat., 1972, p 1449-1458.

7. Bartels, Richard H., Conn, Andrew R., and Sinclair, James W., "Minimization Techniques for Precise Differentiable Functions: The L_1 Solution to an Overdetermined System," SIAM J. Num Anal, (1978) p 224-241.

8. Schlossmacher, E. J., "An Iterative Technique for Absolute Deviations Curve Fitting," JASA, (1973) p 857-859.

9. Daniel, C., and Wood, F.S., "Fitting Equations to Data," Chap. 5, Wiley-Interscience, New York, 1971.

HOW DOES AGEE'S SMOOTHING METHOD WORK?

Carl de Boor
Mathematics Research Center
University of Wisconsin-Madison
Madison, Wisconsin 53706

Bill Agee of White Sands Missile Range has been using a somewhat unusual smoothing method with good success. He told me about it last summer and eventually I began to see what might make it work. This is what I want to explain to you.

First, here is a rough description. of Agee's smoothing method. This description is rough around the edges since I shall not give any indication as to what is happening near the ends of the smoothing interval and also leave out other detail.

Starting with a data point sequence (x_i, y_i), $i=1,\ldots,N$ with $x_i = x_0 + ih$, all i, Agee uses some local smoothing to obtain the smoothed sequence $(x_i, \tilde{y}_i)$, $i=1,\ldots,N$. He then constructs Schoenberg's cubic variation diminishing smoothing spline

$$S := V\tilde{y} := \sum_i \tilde{y}_i C_i$$

(see, e.g., [1, pp. 159ff]). Next, he calculates the difference between $\tilde{y}_i$ and this spline at x_i,

$$e_i := \tilde{y}_i - S(x_i)$$

and applies V to this error sequence, giving him a new spline Ve which he adds to the earlier spline to get a new approximation

$$\hat{S} := S + Ve .$$

He may go through this process once again, getting

$$\hat{\hat{S}} := \hat{S} + V\hat{e} \quad \text{with} \quad \hat{e}_i := \tilde{y}_i - \hat{S}(x_i)$$

and again, getting

$$\hat{\hat{\hat{S}}} := \hat{\hat{S}} + V\hat{\hat{e}} \quad \text{with} \quad \hat{\hat{e}}_i := \tilde{y}_i - \hat{\hat{S}}(x_i)$$

But eventually he stops, with $\hat{S}$ or $\hat{\hat{S}}$ or perhaps even $\hat{\hat{\hat{S}}}$ as his smooth approximation to the original data.

As a first step toward understanding this procedure, consider what would happen if he were to carry out the iteration indefinitely. Think of V as a map applied to a function,

$$Vf := \Sigma_i f(x_i) C_i$$

and, for this purpose, let $\tilde{Y}$ be any function with $\tilde{Y}(x_i) = y_i$, all i. Then the first spline approximation is

$$S^{[1]} = S = V\tilde{Y} ,$$

and, in general,

$$S^{[n+1]} = S^{[n]} + V(\tilde{Y} - S^{[n]})$$

$$= (1 - V)S^{[n]} + V\tilde{Y} , \qquad n=1,2,3,\ldots \quad .$$

In fact, we can start this iteration formula with $n = 0$, provided we set

$$S^{[0]} = 0 .$$

Supposing now $S^{[\infty]} = \lim_{n\to\infty} S^{[n]}$ to exist, it would have to be a fixed point of this iteration, i.e.,

$$S^{[\infty]} = (1 - V)S^{[\infty]} + V\tilde{Y}$$

which then implies that $VS^{[\infty]} = V\tilde{Y}$, or, since the C_i's are linearly independent,

$$S^{[\infty]}(x_i) = \tilde{y}_i \ , \quad \text{all} \ \ i .$$

Conclusion: If the process converges, then it converges to the cubic spline interpolant to the data $(\tilde{y}_i)$.

Since $(\tilde{y}_i)$ is the result of some local smoothing, it would therefore not be a bad idea to use $S^{[\infty]}$. But this is not what Agee does.

In order to understand why Agee's choice of $S^{[3]}$ or $S^{[4]}$ might be better than $S^{[\infty]}$, you have to look now more closely at the iteration map $(1 - V)$. For this, write

$$S^{[n]} =: \sum_i s_i^{[n]} C_i \ .$$

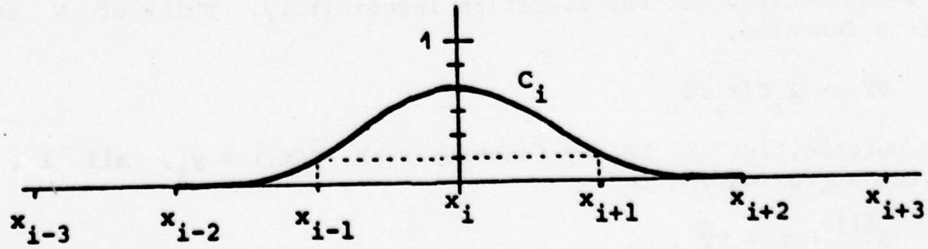
Then the coefficient vectors $s^{[n]}$ are generated by the iteration

$$s^{[n+1]} = (1 - C)s^{[n]} + \tilde{y}$$

with the iteration matrix

$$(1 - C) = (\delta_{ij} - C_j(x_i)) \ ,$$

and I now must tell you more about the functions C_i . These functions are cubic B-splines ($C_i = B_{i-2,4,x}$ in the language of [1]). In particular, C_i looks roughly like



In particular, C_i vanishes off the interval $[x_{i-2}, x_{i+2}]$, and the matrix C, as well as the matrix $1-C$, is therefore tridiagonal. Explicitly, $1 - C$ has the general row

$$-\frac{1}{6} \quad \frac{1}{3} \quad -\frac{1}{6}$$

and has therefore $\|1 - C\|_\infty = 2/3 < 1$. This insures that the iteration process converges. But we need more information.

If you have ever looked at the iterative solution of linear systems, then you will remember that the manner of convergence depends on the spectrum of the iteration matrix. This requires me to bring in one more property of the matrix C, namely the fact that C is totally positive (see, e.g., [1; p. 201]). This means that all minors of C are nonnegative. For our purposes, this has the following consequence:

C has a complete set of eigenvectors $v^{[1]}, \ldots, v^{[N]}$, corresponding to a sequence

$$\lambda_1 > \lambda_2 > \ldots > \lambda_N \geq 0$$

of eigenvalues, all simple and nonnegative. For the particular matrix C, $1 = \lambda_1$ and $\lambda_N > 0$.

With this, expand the solution $x = s^{[\infty]}$ of the linear system $Cx = \tilde{y}$ in terms of the $v^{[i]}$,

$$s^{[\infty]} := \sum_i a_i v^{[i]} .$$

Then, with $s^{[0]} = 0$, we get

$$\begin{aligned} s^{[\infty]} - s^{[k]} &= (1 - C)\,(s^{[\infty]} - s^{[k-1]}) = \ldots \\ &= (1 - C)^k (s^{[\infty]} - s^{[0]}) \\ &= (1 - C)^k s^{[\infty]} = \Sigma_i\, a_i (1 - \lambda_i)^k v^{[i]} \end{aligned}$$

since $(1 - C)v^{[i]} = (1-\lambda_i)v^{[i]}$, hence

$$s^{[k]} = \sum_i [1 - (1 - \lambda_i)^k] a_i v^{[i]} .$$

Correspondingly, we get

$$S^{[k]} = \sum_i [1 - (1 - \lambda_i)^k] a_i V^{[i]} ,$$

with the cubic spline $V^{[i]}$ given by

$$V^{[i]} := \sum_j v_j^{[i]} C_j .$$

This shows in familiar fashion that $S^{[k]}$ approximates the different components $a_i V^{[i]}$ of the interpolant $S^{[\infty]}$ at different rates. For small i, $a_i V^{[i]}$ is already present in $S^{[k]}$ even for small k, since then λ_i is close to 1, hence $(1 - \lambda_i)^k$ goes to zero quite fast as k increases. By contrast, λ_i is close to zero for large i, and this means that $(1 - \lambda_i)^k$ goes to zero only slowly as k increases. Consequently, $S^{[k]}$ has those well approximated only for large k .

This is a good thing if you are trying to smooth because of the following. The total positivity of C also implies that $V^{[i]}$ has exactly $i-1$ sign changes. So, $V^{[1]}$ is constantly equal to 1 , $V^{[2]}$ looks more or less like a straight line, etc. At the other extreme, $V^{[N]}$ changes sign in each interval (x_i,x_{i+1}). But this means that, in the iterative process, the slow moving components of the interpolant $S^{[\infty]}$ are approximated quite quickly, while it takes many iterations to approximate also the fast moving components. This is the reason why it is a good move when smoothing data to stop the iteration after the first few steps, as Agee does.

In conclusion, it is possible to carry out the above analysis without any reference to total positivity since it is easy to construct the vectors $v^{[i]}$ explicitly. But, the above analysis still serves when the data points are not equally spaced in which case there are no explicit or simple formulae for the $v^{[i]}$.

Reference

1. C. de Boor, A practical quide to splines, Springer-Verlag, New York, 1978.

THE GIFT COMPUTER CODE

Gary G. Kuehl

US Army Ballistic Research Laboratory
USA Armament Research and Development Command
Aberdeen Proving Ground, Maryland 21005

ABSTRACT. The Geometric Information for Targets (GIFT) code is a FORTRAN languaged code used to mathematically describe the geometry of a three-dimensional vehicle such as a tank, truck or helicopter. The geometric data generated by the GIFT code is merged in vulnerability computer codes with the energy effects data of a selected munition to simulate the probabilities of malfunction or destruction of components of a vehicle when it is attacked by the selected munition. The GIFT code can be used in the simulation of a portion of a battlefield scenario without actually destroying a costly vehicle.

To simulate the geometry of a vehicle, the GIFT code used combinatorial geometry (COM-GEOM) data. The COM-GEOM data consists of three tables: a Solid Table, a Region Table, and a Region Identification Table. The GIFT code contains numerous tests and checks to ensure that COM-GEOM data listed in these three tables is correct.

The basic computation of the GIFT code involves determining the point of intersection of a ray with a surface. Angles between the ray and the surfaces may also be computed.

The GIFT code options include those which graphically display the vehicle, those which check the correctness of the geometry data, those which compute physical characteristics of the vehicle, and those which generate the geometric data used by vulnerability codes.

1. INTRODUCTION. The Geometric Information for Targets (GIFT) code is a FORTRAN languaged code consisting of approximately 13,000 cards. The GIFT code computes the geometric and physical data associated with a three-dimensional target such as a tank, truck or helicopter. The geometric data generated by the GIFT code is used in vulnerability studies. A vulnerability study consists of an analysis of the probable destructive effects of a weapon on a target. The locations of the target's destroyed or non-destroyed areas from projected views and the ability of the target to move or shoot effectively is addressed in vulnerability studies. The components hit along the trajectory of the projectile(s) are factors considered. The output of the GIFT code provides the components hit along the trajectory of a projectile and a portion of the data required to determine its resistance to penetration.

2. GIFT INPUT. The GIFT code uses combinatorial geometry (COM-GEOM) data to compute the geometric and physical data. COM-GEOM data consists of three tables: a Solid Table, a Region Rable, and a Region Identification Table. The Solid Table contains the basic building blocks of the COM-GEOM data. The Solid Table is a set of geometric solids selected from a group of twelve different geometric codes. These twelve geometric solids are displayed in Figure 1. Note that the figure contains sixteen different solids. The arbitrary convex polyhedron (ARB) depicted is a generic geometric solid that may contain four to eight vertices. The ARS is another generic geometric solid that consists of a large number of planar faces each defined by three points. The ARS solid is normally used to approximate geometric figures such as a casted tank turret that can't be easily described by the other geometric solids.

A number of geometric solids are selectively combined using a series of set theory operations to form the approximate volumetric shape of a portion of a target component called a region in the Region Table. Figure 2 is a two-dimensional representation of the three set theory operations used in the Region Table. The shaded area represents the area described by the operation stated below each pair of overlapping circles.

Each region is described and assigned a code number in the Region Identification Table. Also regions describing air spaces are differentiated from solid components in the Region Identification Table. Figure 3 is an example of a portion of the Region Identification Table. The second column from the right contains a code number denoting the material composition of the region. The far right column contains a percent computed by the volume as it actually exists divided by the volume as it is described. The material and percentage data is not used directly by the GIFT code but is furnished to provide the vulnerability analyst with information helpful in determining a component's resistance to penetration.

The amount of detail used to describe a target is dependent on the vulnerability studies for which it will be used, the amount of time available to generate the data and the size of the computer to be used. A typical low detail description of a tank may contain 300 geometric solids and regions. A typical high detailed description of a tank may contain 1500 solids and regions.

3. ERROR TESTS. The GIFT code employs numerous tests with accompanying error messages to ensure the correctness of the COM-GEOM data. During the execution of the input phase of the GIFT code run, format and geometry error tests of the COM-GEOM data are made. For example, a BOX is described by a vertice and three vectors. One of the geometry error tests is to test that these vectors are normal to each other. During the execution of the option phase of the GIFT code run, overlap tests are made. Overlap tests compare the location of two regions to determine if they are occupying the same space.

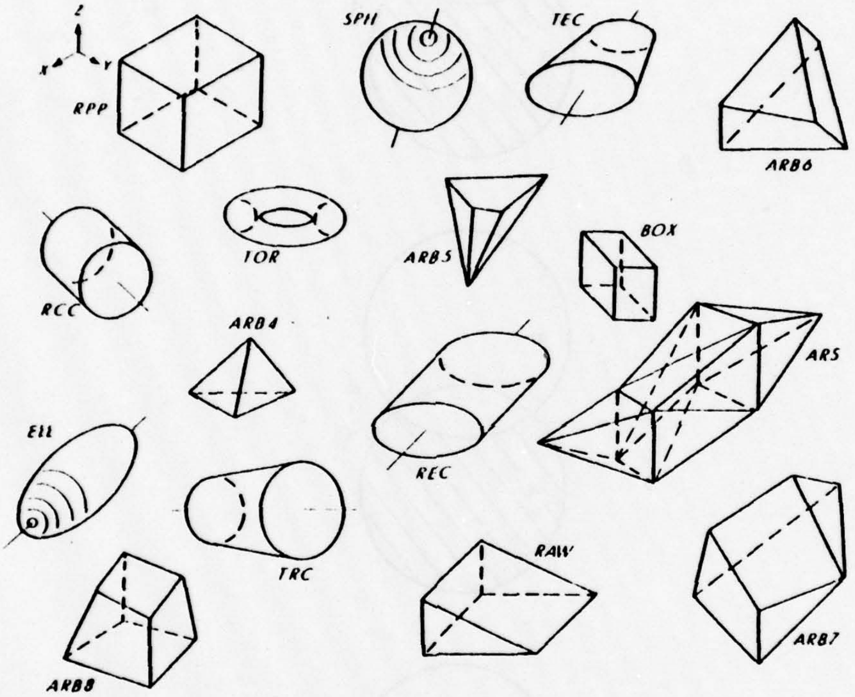


Figure 1. COM-GEOM Solids

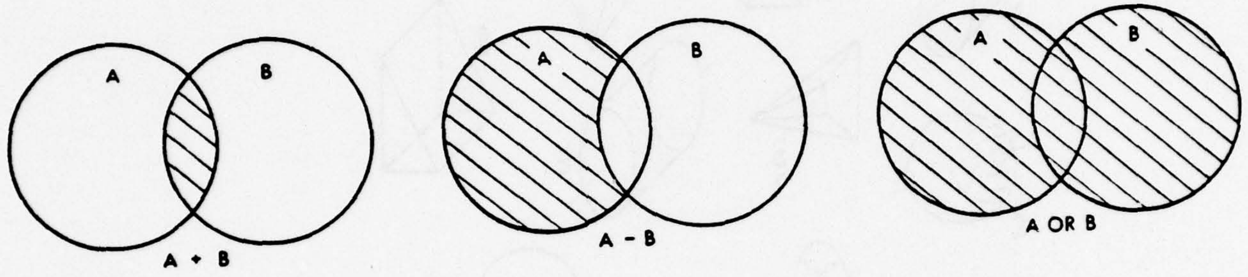


Figure 2. Intersection (+), Subtraction (-), Union (OR) of Solids

REGION	CODE	AIR	DESCRIPTION	MATERIAL CODE	PERCENT
47	108		FLOOR SUPPORT	31	100
48	121		BEAM	31	100
49		4	AIR IN BEAM		
50	104		COMMANDERS HATCH	31	100
51		2	COMMANDERS HATCH AIR		
52	105		COMMANDERS HATCH COVER	31	100
53		2	COMMANDERS HATCH AIR		
54	336		COMMANDERS HATCH HINGE	31	90
55	336		COMMANDERS HATCH HINGE	31	90
56	336		COMMANDERS HATCH HINGE	31	90
57	336		COMMANDERS HATCH HINGE	31	90

Figure 3. An example of a portion of a Region Identification Table expanded for display.

4. BASIC COMPUTATION. The GIFT code's basic computation is determining the distances from the starting point of a ray to the points of intersection of a ray with the surfaces of a geometric solid. The distances to the points of intersection of a ray with all intersecting geometric solids are edited to the level determined by the output option chosen. For example, if the intersection of a ray and Region 1 as displayed in Figure 4 was desired, the distances from the starting point (XB) of the ray (WB) to points C and D would be used in the output. An optional basic computation is determining a vector (WN) normal to a plane tangent at the entrance or exit points or both (C or D or both in example) of the intersection of a region and a ray.

5. GIFT OUTPUT. The GIFT code output options include those used to graphically display the COM-GEOM data, those used to check correctness of the COM-GEOM data, those used to compute the physical characteristics of a target, and those used to generate data for vulnerability codes. Each option consists of one or more subroutines.

The PICTUR, XSECT, and PLTRPP options are used to graphically display the COM-GEOM data. The PICTUR option generates line drawings (hidden lines removed) from any desired view. Although the PICTUR option is basically used for report and display purposes, it may help locate misplaced or misshapened components. Figure 5 displays an example of a line drawing generated by the PICTUR option. One option of PICTUR allows regions to be selectively deleted from the COM-GEOM data and the remaining regions drawn. Figure 6 displays an example using this option with some of the regions located in the upper portion of the target deleted. A box may be optionally described and the portion of the COM-GEOM data located either inside the box or outside the box may be drawn. Figure 7 is an example of this option. The XSECT option generates line drawings or printer plots or both of a intersection of a line with a target. The XSECT drawing is not as pleasing as the PICTUR drawing but the XSECT drawing is computed much faster and provides a truer scaled thickness of a component than the PICTUR drawing. Figure 8 displays an example of a line drawing generated by XSECT. The PLTRPP option plots on printer the approximate location of a selected set of labeled regions.

The CHECK and TESTG options are used to check the correctness of the COM-GEOM data. The CHECK option basically searches the two or more regions occupying the same space and printer plots silhouettes of three basic views (front, side, top) of a region. Figure 10 displays the CHECK output generated for the region depicted in Figure 9. The depicted region is an opened can with a hole in its side. The numbers printed in each silhouette is the number of pairs (one in, one out) of intersections the ray made with the surfaces of the region. The twos (2) in the side view indicates that the can is hollow, and the ones (1) and twos (2) in the top view indicates that the can has a hole. The approximate location

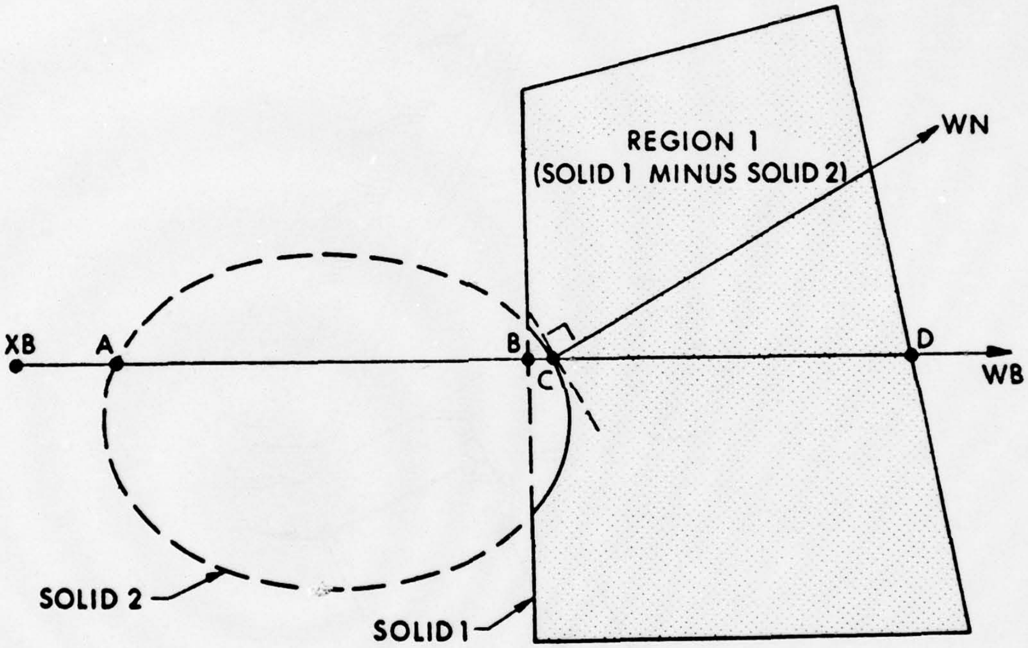


Figure 4. A Ray Intersecting a Region

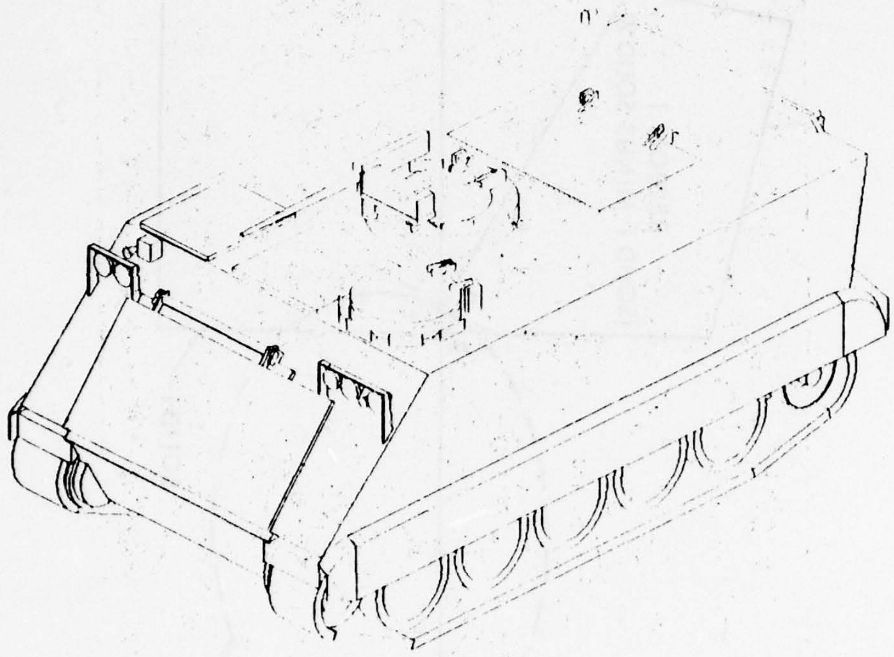


Figure 5. An Example of a PICTUR Plot

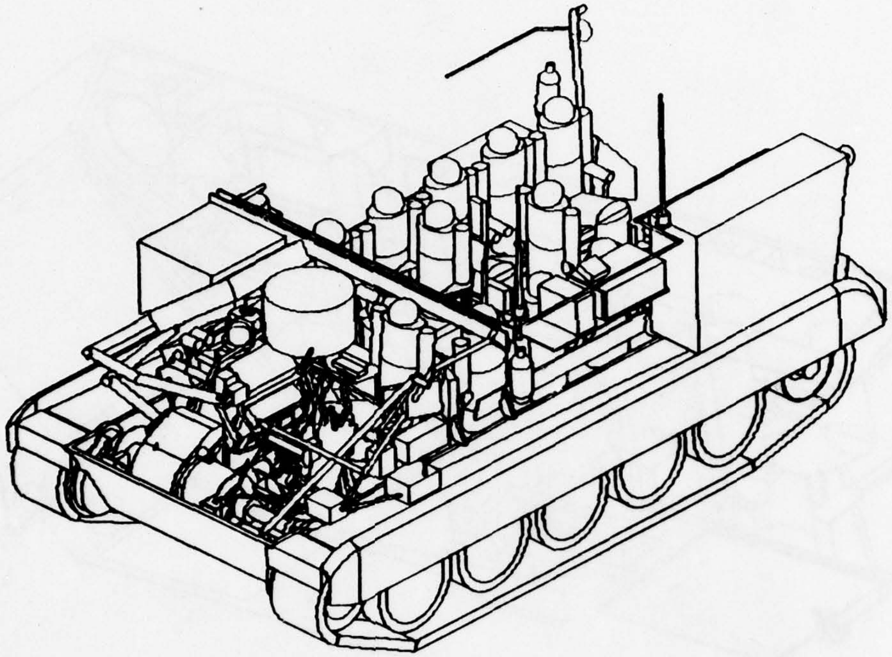


Figure 6. An Example of a PICTUR Plot with Delete Region Option Specified

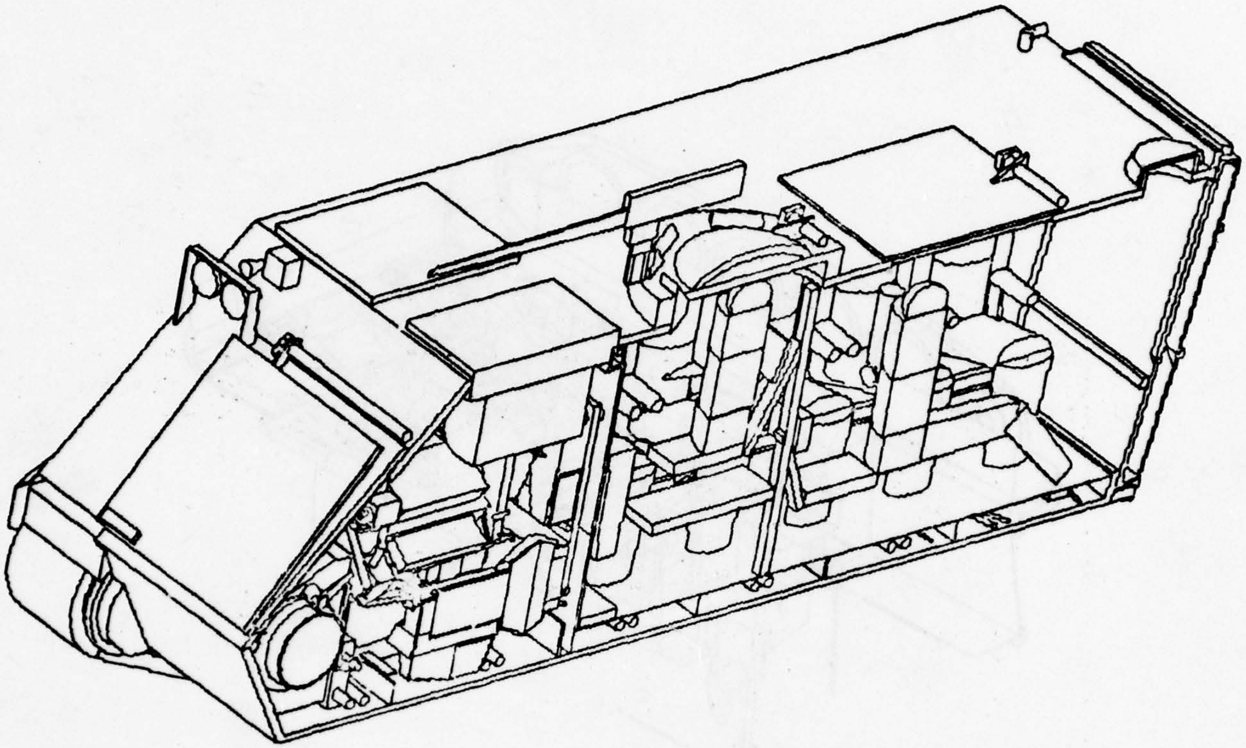


Figure 7. An Example of a PICTUR Plot with Cross Sectional Box Option Specified

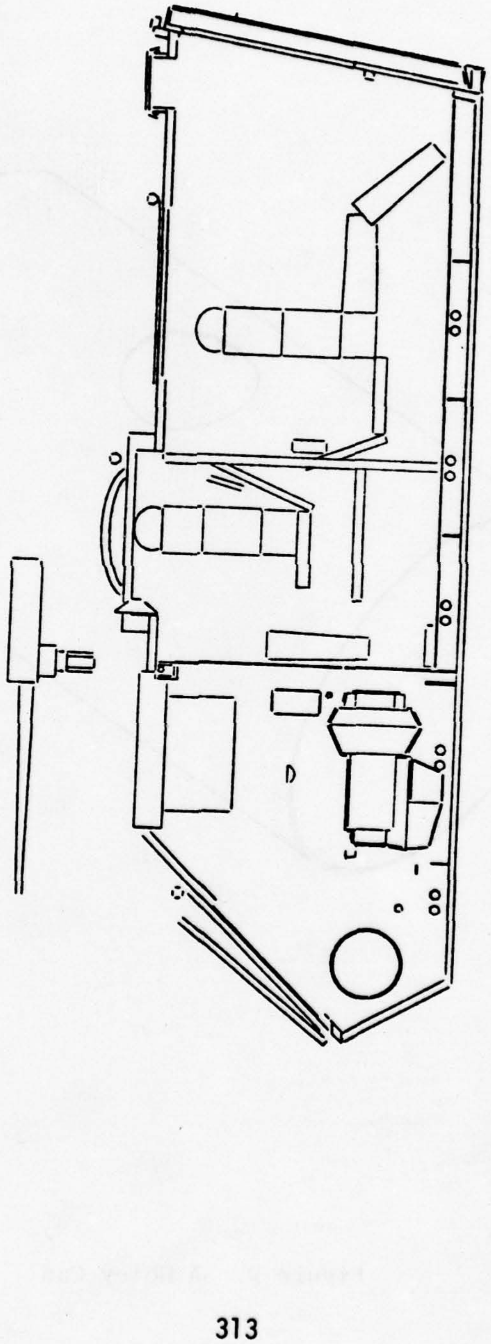


Figure 8. An Example of an XSECT Plot

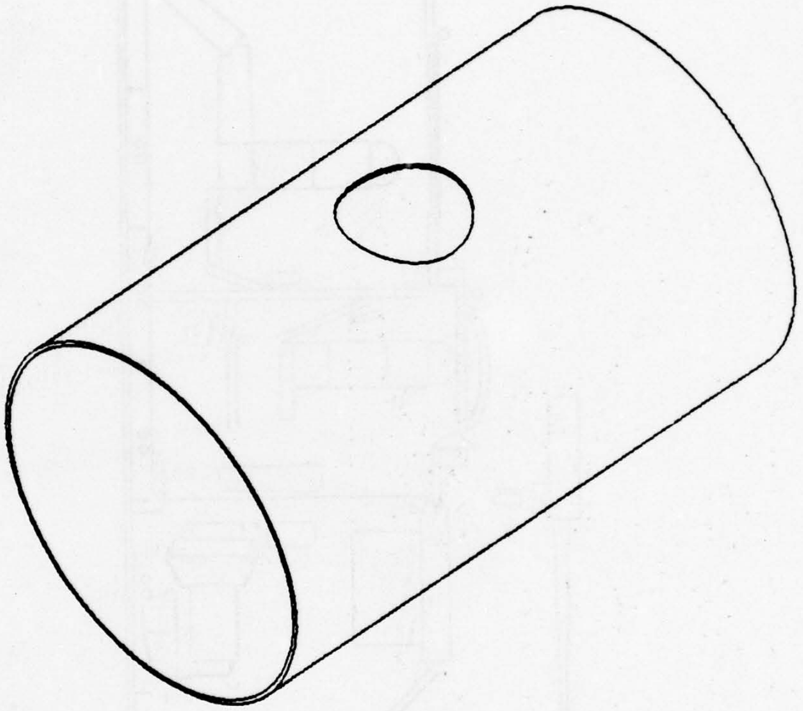


Figure 9. A Holey Can

```
REGION      1    ITEM   100    SPACE   0    DESCRIPTION   HOLEY CAN                                 1 100

HORIZONTAL CELL SIZE       5.40    VERTICAL CELL SIZE       9.00    AREA OF CELL          46.60
XMIN        0.00 XMAX    150.00 XLENGTH    150.00
YMIN      -45.00 YMAX     45.00 YLENGTH     90.00
ZMIN      -45.00 ZMAX     45.00 ZLENGTH     90.00

FRONT
       AREA          VOLUME             SIDE                                        TOP
        6318.0          3159.0                  AREA            VOLUME                      AREA            VOLUME
Y-12345678901234567                            13608.0          18971.4                     13608.0          18576.3
Z                                          X-1234567890123456789012345678             X-1234567890123456789012345678
1      1111111                             Z                                          Y
2    11111111111                           1 2222222222222222222222222222             1 2222222222222222222222222222
3  111111111111111                         2 2222222222222222222222222222             2 2222222222222222222222222222
4  111111111111111                         3 2222222222222222222222222222             3 2222222222222222222222222222
5 11111111111111111                        4 2222222222222222222222222222             4 2222222222222112222222222222
6 11111111111111111                        5 2222222222222222222222222222             5 2222222222211111122222222222
7  111111111111111                         6 2222222222222222222222222222             6 2222222222211111122222222222
8  111111111111111                         7 2222222222222222222222222222             7 2222222222222112222222222222
9    11111111111                           8 2222222222222222222222222222             8 2222222222222222222222222222
0      1111111                             9 2222222222222222222222222222             9 2222222222222222222222222222
                                           0 2222222222222222222222222222             0 2222222222222222222222222222
```

Figure 10. An Example of the CHECK Option Output

of the region, and the presented area and volume of the regīon for each view is also printed. The TESTG option prints as much information as possible about the intersection of a ray and the COM-GEOM data.

The AREA, VOLUME and MOMENT options compute the physical characteristics of a vehicle. The AREA option computes presented areas of regions and components from any desired view. The presented areas of a region or component may or may not optionally include area masked by other components. The VOLUME option computes the volumes and center of points of regions. The MOMENT option computes weights of components and the weight, the moments of inertia and center of gravity of a target when the density for each component is inputted.

The GRID and RIP options generate data used in vulnerability analysis codes. The GRID option generates parallel rays from any desired view and outputs geometric data for each component hit as the ray travels through the target. Figure 11 displays the typical components encountered by a GRID ray. The RIP option generates geometric data that simulates the trajectories associated with a spall producing munition. As a spall producing munition such as an antitank gun penetrates a target, metal fragments (spall) are produced from the breakup of the weapon and the target's armor skin. These metal fragments radiate in a cone from the point of penetration. Figure 12 displays the simulation used by the RIP option. The RIP option generates a (primary) ray that simulates the trajectory of the penetrating weapon. A point (B) is located along this primary ray that lies on the inside surface of the armor (100-199). Rays (spall) simulating the trajectories of the metal fragments are directed toward those (vulnerable) components that can be damaged by fragments from point B. Vulnerable components that lie outside the spall cone are ignored. Regions (shielding) that lie between the vulnerable region and point B along the spall ray are also taken into account.

6. CONCLUSIONS. The GIFT code is a useful tool for doing vulnerability studies. The GIFT makes it possible to do a vulnerability study of a target which isn't available such as a foreign tank or a target that doesn't exist such as a concept vehicle. Once the GIFT's COM-GEOM data is finished, it is very versatile. Not only does it's use in GIFT provide a variety of generated data, but it can also be used in codes akin to GIFT. For example, the same COM-GEOM data could be used in the GIFT code to generate data for a blast overturning vulnerability study or any number of ballistic vulnerability studies. The same COM-GEOM data could be used in codes akin to GIFT in a nuclear radiation vulnerability study or a laser vulnerability study.

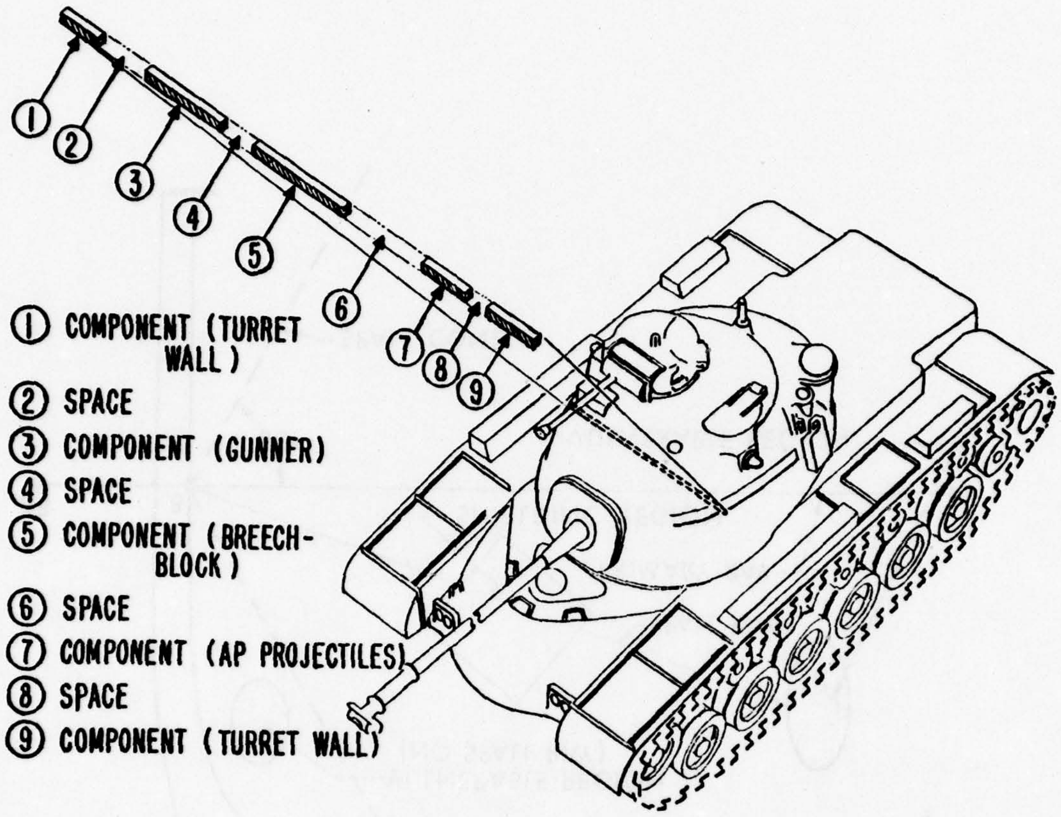

① COMPONENT (TURRET WALL)
② SPACE
③ COMPONENT (GUNNER)
④ SPACE
⑤ COMPONENT (BREECH-BLOCK)
⑥ SPACE
⑦ COMPONENT (AP PROJECTILES)
⑧ SPACE
⑨ COMPONENT (TURRET WALL)

Figure 11. Typical Components Encountered by a GRID Ray

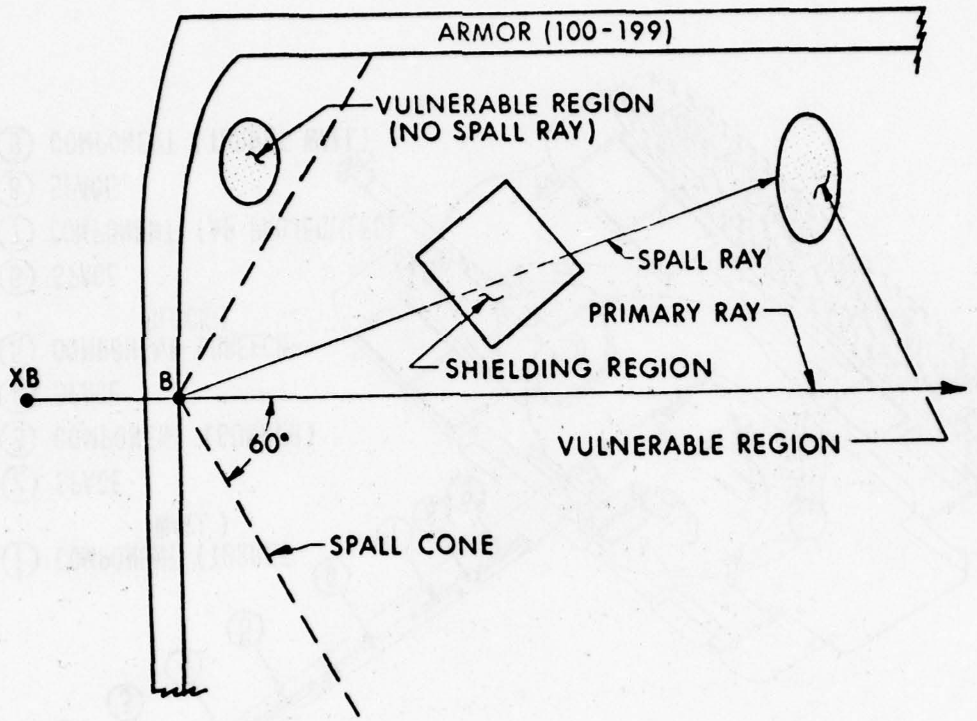


Figure 12. Scheme Used by RIP

RESOLUTION OF AMBIGUITY FOR RESIDUE MEASUREMENT BY ANGLE MEASUREMENT EQUIPMENT

William L. Shepherd and John W. Starner, Jr.
Advanced Technology Office
Instrumentation Directorate
US Army White Sands Missile Range
White Sands Missile Range, New Mexico 88002

ABSTRACT. At US Army White Sands Missile Range (WSMR), there is an electronic angle measurement equipment (AME) system known as Three Object AME (TOAME). The way in which ambiguity occurs in a basic measurement is described and a number-theoretic procedure for using the basic measurements to resolve this ambiguity, is presented. Use of the set of basic measurements to resolve ambiguity when the data is not ideal is also discussed.

Toward the design of AME baselines, some somewhat more general theorems concerning the resolution of ambiguity are presented along with some heuristic statements concerning the numerical stability of the resolution of ambiguity.

1. INTRODUCTION. In section 2 we describe the basic measurement made by AME at WSMR, set down notation, and define ambiguity. Section 3 describes the existing TOAME baseline, in a notation that can be generalized. Resolution of ambiguity for idealized measurements made by TOAME is discussed in section 4. Section 5 presents algorithms for adjusting noisy data, and points out that proper consideration of sensitivity to error is needed in the derivation of algorithms for resolving ambiguity. Sections 6 and 7 present some more theoretical generalizations needed for design of AME baselines.

2. THE BASIC AME MEASUREMENT. Consider Figure 1 X_1 and X_2 are two receiving antennas. α_1 and α_2 are the angles showing the direction from a transmitter, T, to X_1 and X_2. It is supposed that T is distant enough that $\alpha_1 \approx \alpha_2$. Ideally for the algorithms, we would have $\alpha_1 = \alpha_2$, and the incident wavefronts would be planes. In this note we set $\alpha = \alpha_1 = \alpha_2$.

r and ℓ are as indicated in the figure. It can be shown that the phase-meters associated with the X_1X_2 pair of antennas ideally measure a quantity proportional to p, where

$$r \equiv (p + \frac{1}{2})\ mod\ \lambda\ ,\ 0 \leq p \leq \lambda\ , \tag{2.1}$$

and λ is the wavelength of the received transmission. Either of p and $p + \frac{1}{2}$ is variously referred to as a partial, modular, or residue measurement of r. It can be shown that when α is obtuse, r is negative and (2.1) is still valid. (2.1) can also be written

$$r = (i + p + \frac{1}{2})\lambda \quad , \quad i = 0, \pm , \pm 2 , \ldots . \tag{2.2}$$

(2.2) alone is said to give r ambiguously. If there is exactly one (integral) value for i such that the corresponding values for r satisfies

$$- \ell \leq r \leq \ell ,$$

we say that this value for i resolves the ambiguity in r.

It can be shown that if $\ell \leq \frac{1}{2}\lambda$, the ambiguity in r can be resolved. if $\frac{1}{2}\lambda < \ell$, additional information, not contained in the single basic residue measurement, will be needed to resolve the ambiguity. There is no loss in generality in setting $\lambda = 1$, which we will do for convenience.

3. THE TOAME ANTENNA BASELINE. A complete TOAME antenna system consists of four antenna baselines. Associated with each baseline is a set of five antennas, as illustrated in Figure 2. The four baselines, with antennas, are geometrically congruent. The formulas we derive apply to any one of them, as illustrated in Figure 2.

In the ideal case

$$r_j \equiv (p_j + \frac{1}{2}) \; mod \; 1, \tag{3.1}$$

$$- \ell_j \leq r_j \leq \ell_j \quad , \tag{3.2}$$

$$cos \; \alpha = \frac{r_j}{\ell_j} \qquad j = 1, 2, 3, 4, \quad . \tag{3.3}$$

In vector notation, (3.1) can be written

$$r = p' + i, \; i \text{ in } I^4 , \tag{3.4}$$

where I^4 is the set of integral vectors with four components and

$$p' = p + \frac{1}{2}e, \quad e := (1, 1, 1, 1)^T .$$

(3.2) and (3.3) can be written

$$-\ell \geq r \geq \ell , \qquad (3.5)$$

$$r = \ell \cos \alpha . \qquad (3.6)$$

For the TOAME as it is now configured

$$\ell = sc , \qquad (3.7)$$

where $s = .45$ and $c = (36, 42, 43, 288)^T$. (3.8)

Specifically,

$\ell = (16.2, 18.9, 19.35, 129.6)^T$ (wavelengths). (3.9)

The phrase "Angle Measuring Equipment" comes from (3.6). Knowing ℓ and r we can compute $\cos \alpha$ from any component of (3.6). Our ambiguity resolution problem can be stated as:

Given p', find the elements, r, of the set (3.4) which satisfy (3.5) and for which there is a number $\cos \alpha$ such that r and $\cos \alpha$ satisfy (3.6).

It is plausible to suppose that there is at least one such r if p' is an ideal residue measurement of r. In the next section we will show that for TOAME as it is now configured, there is exactly one such r if p' is an ideal residue measurement of r.

4. RESOLUTION OF AMBIGUITY FOR IDEAL RESIDUE MEASUREMENTS BY THE WSMR TOAME. Equations (4.4), (4.5) and (4.1) are adequate for ideal measurements. The rest of this section discusses alternates to be considered, in section 5, for noisy measurements.

Eliminating r and $s \cos \alpha$ from (3.4), (3.6) and (3.7) :

$$Li = -Lp' , \qquad (4.1)$$

$$L = \begin{bmatrix} c_2 & -c_1 & 0 & 0 \\ 0 & c_3 & -c_2 & 0 \\ c_4 & 0 & 0 & -c_1 \end{bmatrix} = \begin{bmatrix} 42 & -36 & 0 & 0 \\ 0 & 43 & -42 & 0 \\ 288 & 0 & 0 & -36 \end{bmatrix} .$$

(4.1) is equivalent to the Diophantine equation

$$Ai = Ap' , \qquad (4.2)$$

$$A = \begin{bmatrix} 7 & -6 & 0 & 0 \\ 2 & 43 & -42 & 0 \\ 8 & 0 & 0 & -1 \end{bmatrix} .$$

Theorems (4.1. a) and (4.2) below state necessary and sufficient conditions on p' so that (4.2) has an integral solution. Theorem (4.1. b) exhibits the set of solutions.

$$B := \begin{bmatrix} -7 & 6 & 0 & 0 \\ 43 & -43 & 6 & 0 \\ -8 & 0 & 0 & 1 \end{bmatrix} .$$

$$a := Bp'$$

$$C := \begin{bmatrix} 1 & 6 & 0 \\ 1 & 7 & 0 \\ 0 & 7 & 0 \\ 8 & 48 & -1 \end{bmatrix} .$$

Theorem 4.1. If (4.2) has an integral solution, then

a. a is an integer.

b. The set of solutions is

$$i = n + cm, \quad m \text{ in } I^1 \qquad (4.3)$$

where

$$n = Ca . \qquad (4.4)$$

A proof of this theorem is omitted. A straightforward but lengthy elementary proof is given in reference [1].

Theorem 4.2. If a is an integer then (4.2) has an integral solution.

Proof: From the definitions of n and C, n is an integer. n = Ca = CBp'. Direct substitution of CBp' to n into (4.2) and simplification shows that n is a solution of (4.2).

We can now say that, if a is an integer, then n is an integer and (by using (3.4), (3.6) and (3.7)) that r is in the set

$$r = p' + n + cm \;, \quad m \text{ in } I^1 . \tag{4.5}$$

We now present some alternates to (4.4) and some alternate necessary and sufficient conditions.

$$H := \begin{bmatrix} 7 & -6 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -48 & 48 & 0 & -1 \end{bmatrix} .$$

It can be verified that

$$HC = I \; . \tag{4.6}$$

Multiplying (4.4) by H and using (4.6) ,

$$a = Hn \; .$$

We now have the theorem that n is an integer if and only if a is integer. Further, we can write

$$n = CBp' = Gp' \quad , \tag{4.7}$$

$$G = \begin{bmatrix} 251 & -252 & 36 & 0 \\ 294 & -295 & 42 & 0 \\ 301 & -301 & 42 & 0 \\ 2016 & -2016 & 288 & -1 \end{bmatrix} .$$

$$D: = \begin{bmatrix} 1 & 0 & -1 \\ 7 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix} .$$

$$b: = Da = DBp' = Ep' \; , \tag{4.8}$$

$$E = \begin{bmatrix} 1 & 6 & 0 & -1 \\ 2 & -1 & 6 & -1 \\ -8 & 0 & 0 & 1 \end{bmatrix} .$$

It can be verified that

$$a = D^{-1}b ,$$

$$D^{-1} = \begin{bmatrix} 1 & 0 & 1 \\ -7 & 1 & -6 \\ 0 & 0 & 1 \end{bmatrix} .$$

so b is an integer if and only if a is integer.

$$n = Fb , \tag{4.9}$$

$$F = \begin{bmatrix} -41 & 6 & -35 \\ -48 & 7 & -41 \\ -49 & 7 & -42 \\ -328 & 48 & -281 \end{bmatrix} .$$

Returning to ambiguity resolution, we want to know the elements of (4.5) which satisfy (3.5). From (3.5), (3.7) and (4.5) ,

$$-\beta c \le p' + n + cm \le \beta c ,$$

$$- p' - n - \beta c \le cm \le - p' - n + \beta c \quad .$$

$$\frac{-p_j' \ -n_j}{c_j} - \beta \le m \le \frac{-p_j' \ \ -n_j}{c_j} + \beta \ , \ j = 1, 2, 3, 4 \ . \tag{4.10}$$

From (4.10), r is in $[-\ell, \ell]$ if and only if m is in

$$I_j = [\frac{-p_j' - n_j}{c_j} - \beta \ , \ \frac{-p_j' - n_j}{c_j} + \beta] \quad . \tag{4.11}$$

since the length of I_j is $2\beta = .9 < 1$, there is at most one integer in I_j. For ideal measurements there is an r in $[-\ell, \ell]$, or an integer, $\hat{m}$, in I_j and $\hat{m}$ solves the ambiguity problem.

5. APPROXIMATE AMBIGUITY RESOLUTION FOR NOISY RESIDUE MEASUREMENTS. In this section we still suppose that the transmitter is distant enough from the AME baseline that the wavefronts incident on the baseline are planar, but introduce the notion of noisy measurements. Three methods for approximate resolution of ambiguity for noisy data are presented and compared. p' is now considered to be a given noisy vector, with

$$\frac{1}{2}e \leq p' \leq \frac{3}{2}e \, . \tag{5.1}$$

Again consider

$$Ai = -Ap' \, . \tag{5.2}$$

It is not likely that (5.2) has an (integral) solution. We use the necessary and sufficient conditions of section 4 to obtain the three approximate solutions.

From section 4, a (not necessarily integral) solution of (5.2) is

$$n := Gp' = Ca = Fb \, . \tag{5.3}$$

If any of n, a or b is an integer

$$i = n + cm, \quad m \text{ in } I'$$

is the set of integral solutions. With $[X]_I$ = integer nearest X, (5.4), (5.5) and (5.6) give three easily computed integral approximations, $\hat{n}$, to n, each of which is exact if n is an integer:

$$\hat{n} = [Gp']_I \tag{5.4}$$

$$\hat{n} = C\hat{a} \, , \qquad \hat{a} = [B_{p'}]_I \quad , \tag{5.5}$$

$$\hat{n} = F\hat{b} \qquad \hat{b} = [Ep']_I \quad , \tag{5.6}$$

For each of these,

$$\hat{i} = \hat{n} + cm \qquad m \text{ in } I' \tag{5.7}$$

is a set of integers approximately satisfying (5.2) .

Define, for a matrix, M,

$$||M|| = \max_i \sum_k |m_{i\,k}| .$$

We use $||M||$ as a measure of sensitivity to error in x of the computation of Mx.

It is easily verified that $||G|| = 4417$, $||B|| = 92$, $||E|| = 10$. Since $10 < 92 < 4417$ we choose (5.6) and (5.7) as a set of approximate integral solutions. We do not know that this is the "best" possible such set.

In order to complete the resolution of ambiguity, consider tne set S, of real numbers, x, such that $Ex = \hat{b}$.

Define $\hat{p}'$, in S, by

$||\hat{p}' - p'|| \leq ||x - p'||$, for all x in S.

$$I_j = [- \frac{\hat{p}_j + \hat{n}_j}{cj} - \beta \; , - \frac{\hat{p}_j + \hat{n}_j}{cj} + \beta]$$

Define $\hat{m}$ to be the integer nearest I_j.

$\hat{i}: = \hat{n} + C\hat{m}, \; \hat{n} = F\hat{b}.$

$\hat{r}: = \hat{i} + \hat{p}'.$

It can be shown that $\hat{i}$ is a solution of

$$Ai = - Ap' .$$

However, $\hat{r}$ is in $[-\ell, \ell]$ if and only if $\hat{m}$ is in I_j. In either case we say that $\hat{m}$ approximately resolves the ambiguity in $\hat{r}$.

Finally it can be shown, much as on page 227 of reference [2], that

$$\hat{p}' = p' + E^T (EE^T)^{-1}(\hat{b} - b) \; , \qquad (5.8)$$

$$E^T(EE^T)^{-1} = \frac{1}{87843} \begin{bmatrix} -295 & -258 & -10291 \\ 14298 & -301 & 1901 \\ 2088 & 14334 & 4038 \\ -2360 & -2064 & 485 \end{bmatrix} .$$

From the preceding lengthy discussion, the following rather simple formulas can be extracted. From them can be constructed simple and time-efficient computer programs for the approximate resolution of ambiguity in r (or $cos\ \alpha$), given the noisy residue measurement, p.

$$p' = p + \frac{1}{2}e, \quad e = (1, 1, 1, 1)^T . \tag{5.9}$$

$$b = \begin{bmatrix} 1 & 6 & 0 & -1 \\ 2 & -1 & 6 & -1 \\ -8 & 0 & 0 & 1 \end{bmatrix} p'. \tag{5.10}$$

$$\hat{b} = \text{integer nearest } b. \tag{5.11}$$

$$\hat{n}_4 = (-328 \quad 48 \quad -281)\hat{b} \tag{5.12}$$

$$\hat{p}'_4 = \frac{1}{87853}(-2360 \quad -2064 \quad 485)(\hat{b} - b) + p'_4. \tag{5.13}$$

$$\hat{m} = \text{integer nearest } [-\frac{\hat{n}_4 + \hat{p}'_4}{288} - .45, -\frac{\hat{n}_4 + \hat{p}'_4}{288} + .45]. \tag{5.14}$$

$$\hat{r}_4 = \hat{n}_4 + \hat{p}'_4 + \hat{m}. \tag{5.15}$$

$$cos\ \alpha \simeq \frac{\hat{r}_4}{129.6} . \tag{5.16}$$

6. THEORETICAL CONSIDERATIONS . The work discussed so far is based on an analysis of a specific set of baseline measurements. The antenna spacings are shown to be adequate to resolve ambiguity, and it is shown how, for a reasonable set of residue measurements, to recover the correct integer vector i. An interesting question arises. Is there a set of conditions we can place on the set of baseline lengths so that we can resolve ambiguity and recover the correct integer vector i for reasonable noise in the data measurements? If such a set of conditions can be stated we can design useable baselines. In this section we will in fact present such a set of conditions.

Consider the baseline with n antenna spacings (n+1 antennas). There are many ways to eliminate r *cos* α and β from the equations as was done in (4.1). The following is a generalizable way. Define L as

$$L = \begin{bmatrix} c_2 & -c_1 & & 0 \\ & c_3 & -c_2 & \\ 0 & & c_n & -c_{n-1} \end{bmatrix} .$$

The system of equations can be written Li = -Lp'. Theorem 6.1 will show us how to choose the c_i so that we have a necessary and sufficient condition that an integral solution to Li = -Lp' exists.

Theorem 6.1. Let the vector c be defined as above (3.7), with integral components. Further let the n-1 values $c_1/(c_1, c_2)$, $c_1/(c_1, c_3)$, ..., $c_1/(c_1, c_n)$ (where (a,b) is the greater and common divisor of a and b) be relatively prime in pairs. Let B be the matrix

$$\begin{bmatrix} \frac{c_2}{(c_1\ ,\ c_2)} & -\frac{c_1}{(c_1\ ,\ c_2)} & 0 & \\ \frac{c_3}{(c_1\ ,\ c_3)} & 0 & -\frac{c_1}{(c_1\ ,\ c_3)} & \\ \vdots & & & \\ \frac{c_n}{(c_1\ ,\ c_n)} & \cdot\ \cdot & 0 & -\frac{c_1}{(c_1\ ,\ c_n)} \end{bmatrix} .$$

There is an integral solution i to Li = -Lp' if and only if Bp' is integral.

Proof. The matrix B is obtained from the matrix L by elementary row transformation, so Bi = -Bp' has an integral solution if and only if Li = -Lp' has an integral solution. Clearly if i is integral then Bp' is integral. Assume Bp' is integral and show there exists an integral solution i. Bi = -Bp' can be written

$$c_2/(c_1, c_2)\, i_1 = (-Bp')_1 + c_1/(c_1, c_2)\, i_2$$

$$c_3/(c_1, c_3)\, i_1 = (-Bp')_2 + c_1/(c_1, c_3)\, i_3$$

$$\vdots \qquad \vdots \qquad \vdots$$

$$c_n/(c_1, c_n)\, i_1 = (-Bp')_n + c_1/(c_1, c_n)\, i_n$$

It is not difficult to show that since the numbers $c_1/(c_1, c_2)$, $c_1/(c_1, c_3)$, ..., $c_1/(c_1, c_n)$ are relatively prime in pairs then the Chinese remainder theorem guarantees the existence of an integer i that satisfies each of the equations. Therefore an integer exists so that Bi = -Bp' where -Bp' is integral. The next theorem further restricts the choice of the c vector and restricts the value of β in order that a unique solution exists.

Theorem 6.2. In addition to the restrictions placed on c in theorem 6.1 let the components of c form a relatively prime set. Further let β be so that $0< \beta \leq .5$. If p' is given so that an integral solution i to Li = -Lp' exists for which $-\ell \leq r \leq \ell$, then that solution is unique.

Proof. Let i be a solution Li = -Lp' so that $-\ell \leq r \leq \ell$. Clearly L is of rank n-1. The null space of L is one dimensional; and any nonzero element of this space is a basis for it. By inspection we see that Lc = 0, i.e., c is in the null space of L. Therefore any solution to Li = -Lp is of the form i + αc. Since the components of c are a relatively prime set α must be integral. Successive integral solutions are separated componentwise by c. Since $\ell = \beta c$ and $\beta \leq \frac{1}{2}$ the integral solutions are separated componentwise by at least 2ℓ. There can be at most one solution in the interval $[-\ell,\ell]$.

7. DESIGN CONSIDERATIONS. In this section we consider design in terms of sensitivity of our ability to recover the proper integer value with respect to noisy data. In section 5 we obtained 3 approximations for the value $\hat{n}$ (5.4, 5.5, 5.6). Each of these was obtained by rounding some vector componentwise to the nearest integer vector. We will do the same thing here. Let p' be a set of noisy data values. Bp' will not in general be integral. Let $\hat{a} = [Bp']_I$ be integral as in section 5. Our objective here is to modify B and also the baseline in such a way as to maintain the properties of theorems 6.1 and 6.2 and reduce $||B||$ as defined in section 5.

The first thing we do is add another antenna to the system. The new spacing ℓ_{n+1} is chosen as some integral multiple γ of ℓ_k for some k, where the set $c_1, c_2, \ldots, c_{n+1}$ is still a relatively prime set. The values γ and k will be chosen later to help reduce $||B||$. The new B matrix is $\bar{B}$,

$$\bar{B} = \begin{bmatrix} \frac{c_2}{(c_2,\, c_2)} & -\frac{c_1}{(c_1,\, c_2)} & & \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ \frac{c_n}{(c_1,\, c_n)} & & & -\frac{c_1}{(c_1,\, c_n)} \\ 0 & & \gamma & -1 \end{bmatrix} .$$

Let $\bar{i}$ and $\bar{p}'$ be the new i and p' vectors appropriately augmented. It can be easily seen by partitioning the new problem $\bar{B}\bar{i} = -\bar{B}\bar{p}'$ and applying theorem (6.1) that there is an integral solution i if and only if $\bar{B}\bar{p}'$ is integral. Further since the set $c_1, c_2, \ldots, c_{n+1}$ is relatively prime the solution is unique in $[-\ell, \ell]$.

We can further modify B by taking carefully chosen row combinations. Let P be any permutation matrix and T be triangular with integral elements and unit diagonal. Since $(TP)^{-1} = P^{-1}T^{-1}$ is integral TP $\bar{B}P'$ is integral if and only if $\bar{B}p'$ is. Choose T, P, and γ and k so that the norm of TP$\bar{B}$ is reduced. This can be seen by example. Let the original c vector be $(42, 63, 66, 70)^T$. The B matrix is

$$B = \begin{bmatrix} 3 & -2 & 0 & 0 \\ 11 & 0 & -7 & 0 \\ 5 & 0 & 0 & -3 \end{bmatrix}.$$

Let $\gamma = 3$, $k = 4$, then the $\bar{B}$ matrix is

$$\bar{B} = \begin{bmatrix} 3 & -2 & 0 & 0 & 0 \\ 11 & 0 & -7 & 0 & 0 \\ 5 & 0 & 0 & -3 & 0 \\ 0 & 0 & 0 & 3 & -1 \end{bmatrix}.$$

Let P = I and T be given

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -2 & -2 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Then TP$\bar{B}$ is

$$TP\bar{B} = \begin{bmatrix} 3 & -2 & 0 & 0 & 0 \\ 1 & 0 & -7 & 0 & 2 \\ 5 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 3 & -1 \end{bmatrix}.$$

The norm of the original B is 18. The norm of TP$\bar{B}$ is 10. This is not a great reduction, but it shows the principle behind the choice of T, P, γ and k. The original B matrix for this baseline already had fairly small norm. Using the TP$\bar{B}$ matrix an appropriate left inverse can be computed and the algorithm for solution can be set up as in section 5.

REFERENCES

[1] "Resolution of Ambiguity in the Three Object Angle Measuring Equipment (TOAME) at White Sands Missile Range", William L. Shepherd and John W. Starner, Jr. Advanced Technology Research Memorandum 78-11.

[2] "Modular Measurements of a Distance", William L. Shepherd SIAM Review, Vol. 9, No. 2, pp. 223 through 228.

$$r - \tfrac{1}{2}\lambda = p + i\lambda$$

$$0 \leq p < \lambda$$

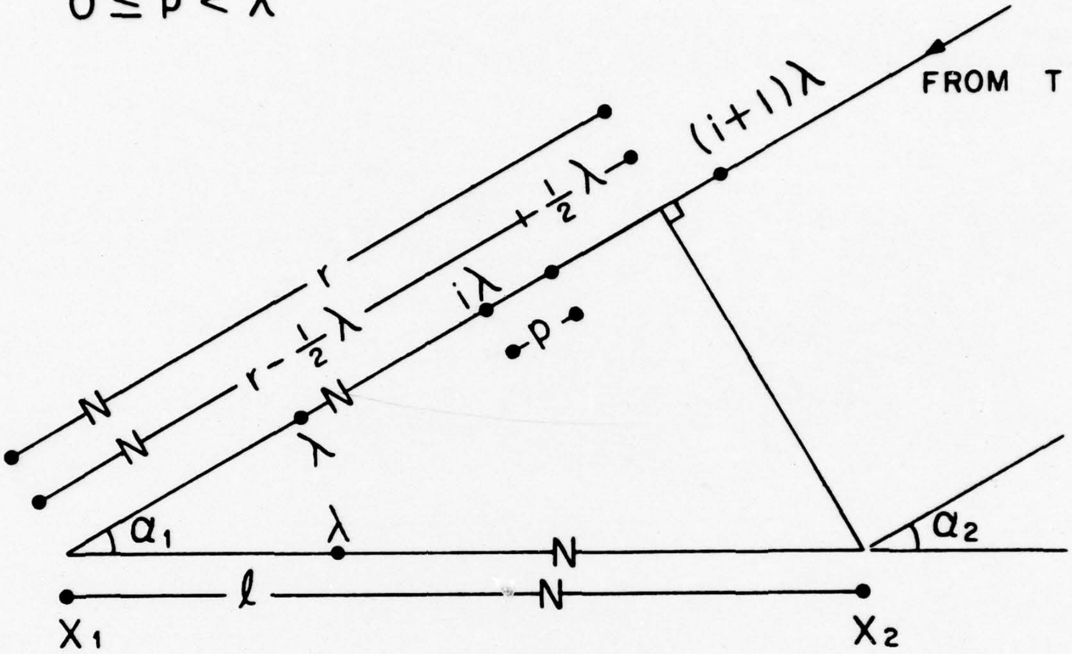


FIGURE 1. BASELINE FOR AN ANTENNA PAIR

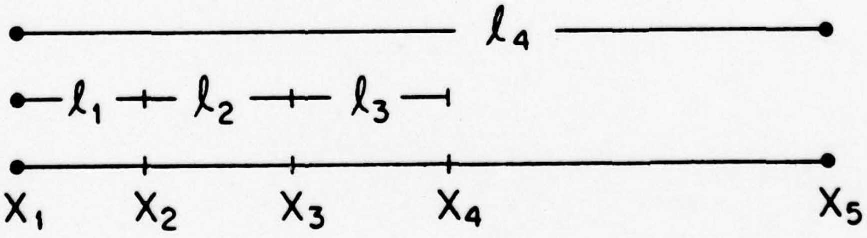


FIGURE 2. TOAME BASELINE ANTENNA HOOK-UP

A COMPARISON OF THE EXISTENCE THEOREMS OF KANTOROVICH AND MOORE

L. B. Rall
Mathematics Research Center
University of Wisconsin - Madison
610 Walnut Street
Madison, Wisconsin 53706

ABSTRACT. In order to be useful, an approximate solution y of a nonlinear system of equations $f(x) = 0$ in R^n must be close to a solution x^* of the system. Two theorems which can be used computationally to establish the existence of x^* and obtain bounds for the error vector $y - x^*$ are the 1948 result of L. V. Kantorovich and the 1977 interval analytic theorem due to R. E. Moore. The two theorems are compared on the basis of sensitivity (ability to detect a solution x^* close to y), precision (ability to give sharp error bounds), and computational complexity (cost). A theoretical comparison shows that the Kantorovich theorem has at best only a slight edge in sensitivity and precision, while Moore's theorem requires far less computation to apply, and thus provides the method of choice. This conclusion is supported by a numerical example, for which available UNIVAC 1108/1110 software is used to check the hypotheses of both theorems automatically, given y and f.

1. Nonlinear Systems of Equations.

A system of n nonlinear algebraic or transcendental equations in n real unknowns,

$$(1.1)\qquad \begin{cases} f_1(x_1,x_2,\dots,x_n) = 0 , \\ f_2(x_1,x_2,\dots,x_n) = 0 , \\ \dots \quad \dots \quad \dots \quad \dots \\ f_n(x_1,x_2,\dots,x_n) = 0 , \end{cases}$$

may be represented concisely in the real n-dimensional vector space R^n as the equation

$$(1.2)\qquad f(x) = 0 ,$$

where $f\colon D \subset R^n \to R^n$ is a nonlinear operator from a domain $D \subset R^n$ into R^n. The problem of solving the system (1.1) or the equivalent equation (1.2) is, of course, to find a *solution* $x^* = (x_1^*,x_2^*,\dots,x_n^*) \in D$ which f maps into the origin $0 = (0,0,\dots,0)$ of R^n.

In order to keep the discussion of this problem within the realm of practical computation, it will be assumed that the functions $f_i(x_1,x_2,\dots,x_n)$, $i = 1,2,\dots,n$, comprising the components of $f(x)$ can be written as formulas in FORTRAN or some similar computer language. As software exists for analytic

Sponsored by the United States Army under Contract No. DAAG29-75-C-0024.

differentiation of functions of this type [4,9], it will also be assumed without further ado that the derivatives $f'_{ij} = \partial f_i/\partial x_j$ and, if necessary, $f''_{ijk} = \partial^2 f_i/\partial x_j \partial x_k$ can be computed automatically, so that, in particular, the evaluation of the Jacobian matrix

(1.3) $$f'(x) = \left(\frac{\partial f_i(x)}{\partial x_j}\right)$$

presents no difficulty for $x \in D$.

2. Existence theorems and error bounds. Actual computation with one of the many methods for solving systems of equations will yield an *approximate solution* y, which will be useful if it can be asserted that y is "close" to a solution x^* of (1.2). This involves establishing (i) the *existence* of x^* in some region Ω^* containing y, and (ii) some type of *bound* for the *error vector* $\epsilon = x^* - y$ or its components.

In the case of linear systems $Ax = b$, verification of existence rests on a finite number of aritimetic operations. For linear systems of reasonable size, errors in computed approximate solutions result only from rounding, and successful execution of a carefully written program can insure the existence of x^* and provide usable error estimates [2]. However, in the nonlinear case, such assurance of existence may be lacking. Furthermore, even if known to exist, x^* may be defined only as the limit of an infinite process, in which case y will differ from x^* due to truncation as well as roundoff error.

Some common criteria for accepting y as a good approximation to x^* are (i) the *residual*

(2.1) $$r = f(y)$$

is small, or (ii) the *correction*

(2.2) $$\delta_k = x^{(k+1)} - x^{(k)}$$

is small when calculating x^* as the limit of a sequence $\{x^{(k)}\}$, in which case one takes $y = x^{(k+1)}$. Either of these can fail for nonlinear f in R. For example, if $f(x) = e^{-x}$, then the residual r can be made arbitrarily small, but x^* does not exist such that $f(x^*) = 0$. Similarly, an iteration process $x^{(k+1)} = \phi(x^{(k)})$ can stall far from x^*. To see this, apply Newton's method to (1.2) with $f(x) = x^m$ and $x^{(0)} = 1$. The correction δ_0 can be made as small as desired by taking m sufficiently large; the value obtained for $y = x^{(1)}$ will be close to 1 and thus not a good approximation to $x^* = 0$.

It follows that there is a need for *computational* existence theorems such that given y and f; their hypotheses can be checked by a computer program, and error bounds obtained if existence is verified. Other desirable properties for such theorems are:

(i) Sensitivity. The region Ω^* in which the theorem can detect the existence of a solution x^* of equation (1.2) is as large as possible.

(ii) Precision. The region Ω^* in which x^* is guaranteed to exist does not extend far beyond x^*, so that the error bounds obtained are as good as possible.

(iii) Simplicity. The additional computation required to guarantee existence and obtain error bounds should be as inexpensive as possible.

To a certain extent, sensitivity and precision are incompatible, at least in a single theorem. A sensitive theorem might establish existence of x^* in a large region, but not yield usable error bounds. A result requiring highly precise approximate solutions, on the other hand, might fail to detect a solution x^* which is close enough to make the accuracy of y satisfactory for the intended purpose. Consequently, some compromise between sensitivity and precision must be struck. Of course, a computational strategy can be devised in which a sensitive theorem is used to scout a large region for suitable initial approximations, which are then refined until a precise theorem guarantees sufficient accuracy [13].

The purpose of this paper is to compare two computational existence theorems, for which UNIVAC 1108/1110 software is operational, on the basis of sensitivity, precision, and simplicity.

3. The theorems of Kantorovich and Moore. Theorems which can be implemented computationally to obtain verifiable conditions for existence of solutions as well as error bounds include the well-known result on the convergence of Newton's method due to L. V. Kantorovich [6] and the more recent interval-analytic theorem of R. E. Moore [12]. Brief statements of these theorems will now be given; proofs may be found in the literature. In particular, a neat proof of the form of the Kantorovich theorem presented here may be found in the note by Ortega [14].

Let $\|\cdot\|$ denote a norm for R^n and also a consistent matrix norm for $n\times n$ real matrices; that is, $\|Ax\| \le \|A\|\cdot\|x\|$ for all $x \in R^n$ and square matrices A of order n with real elements. The ingredients of the Kantorovich theorem are (i) an *initial point* (approximate solution) $x^{(0)}$ at which the Jacobian matrix $f'(x^{(0)})$ is invertible, with

$$(3.1) \qquad \|[f'(x^{(0)})]^{-1}\| \le B_0 ,$$

and (ii) a Lipschitz constant κ for f' such that

$$(3.2) \qquad \|f'(u) - f'(v)\| \le \kappa\|u-v\|, \quad u,v \in \Omega ,$$

where Ω is a sufficiently large region containing $x^{(0)}$. (The meaning of "sufficiently large" will be made precise below.) From (i), the *Newton point*

$$x^{(1)} = x^{(0)} - [f'(x^{(0)})]^{-1}f(x^{(0)}) \tag{3.3}$$

is uniquely defined, and one can find a constant η_0 such that

$$\|x^{(1)} - x^{(0)}\| \leq \eta_0 . \tag{3.4}$$

Theorem 3.1 (Kantorovich). If

$$h_0 = B_0 \kappa \eta_0 \leq \frac{1}{2} , \tag{3.5}$$

and $\bar{U}(x^{(0)},\rho_0) = \{x : \|x-x^{(0)}\| \leq \rho_0\} \subset \Omega$ for

$$\rho_0 = \frac{1 - \sqrt{1-2h_0}}{h_0} \eta_0 , \tag{3.6}$$

then there exists a solution $x^* \in \bar{U}(x^{(0)},\rho_0)$ of equation (1.2).

The conclusion of this theorem provides a guarantee of the existence of a solution x^* and also the error bound

$$\|x^* - x^{(0)}\| \leq \rho_0 \tag{3.7}$$

for $x^{(0)}$ as an approximate solution of equation (1.2). Using the Newton point $x^{(1)}$ instead of $x^{(0)}$ as the approximate solution, one can obtain the sharper error bound

$$\|x^*-x^{(1)}\| \leq \frac{1-h_0- \sqrt{1-2h_0}}{h_0} \eta_0 , \tag{3.8}$$

as shown by Gragg and Tapia [3]. In practice, one may take $\Omega = \bar{U}(x^{(0)},2\eta_0)$, as if κ is the Lipschitz constant for this region, then $\bar{U}(x^{(0)},\rho_0) \subset \bar{U}(x^{(0)},2\eta_0) = \Omega$ if and only if $h_0 \leq \frac{1}{2}$.

Moore's theorem is based on the concepts of interval analysis [11]. Given vectors $a = (a_1,a_2,\ldots,a_n)$ and $b = (b_1,b_2,\ldots,b_n)$ in R^n with $a_i \leq b_i$, $i = 1,2,\ldots,n$, the *interval* $X = [a,b]$ in R^n is defined to be

$$X = [a,b] = \{x:a_i \leq x_i \leq b_i,\ i = 1,2,\ldots,n\}. \tag{3.9}$$

The *interval extension* G of a continuous function $g:D \subset R^n \to R^n$ may be constructed on intervals $X \subset D$ in the following way: For $g(x) = (g_1(x), g_2(x),\ldots, g_n(x))$, take

$$c_i = \min_{u \in X} g_i(u) , \quad d_i = \max_{v \in X} g_i(v), \quad i = 1,2,\ldots,n , \tag{3.10}$$

and define $G(X) = [c,d]$. It follows that $G(X) \supset \{g(x):x \in X\} = g(X)$; however, $G(X)$ may also contain vectors which are not of the form $g(x)$ for some $x \in X$.

The recipe for Moore's theorem [12] calls for an initial point y, an interval X containing y, and a nonsingular real matrix Y. These are used to form the *Krawczyk transformation*

$$K(X) = y - Yf(y) + \{I-YF'(X)\}(X-y) \tag{3.11}$$

of the interval X, where I is the identity matrix, and F' denotes the interval extension of the derivative f' of f.

Theorem 3.2 (Moore). If

$$K(X) \subset X, \tag{3.12}$$

then there exists a solution $x^* \in K(X)$ of equation (1.2).

This theorem also provides error bounds in addition to a guarantee of the existence of a solution. Setting $K(X) = [c,d]$, one obtains the componentwise error bounds

$$|x_i^*-y_i| \leq \max\{|y_i-c_i|,|d_i-y_i|\}, \quad i = 1,2,\ldots,n, \tag{3.13}$$

for y as an approximate solution of equation (1.2). Error bounds analogous to (3.8) may be obtained by setting

$$w = y-Yf(y), \quad [s,t] = \{I-YF'(X)\}(X-y), \tag{3.14}$$

from which

$$|x_i-w_i| \leq t_i-s_i, \quad i = 1,2,\ldots,n, \tag{3.15}$$

as $(x-w) \in [s,t]$.

Although different in appearance, Theorems 3.1 and 3.2 have a common background. Define the *Newton operator* ϕ by

$$\phi(x) = x - [f'(x)]^{-1}f(x), \tag{3.16}$$

and the *Newton sequence* $\{x^{(k)}\}$ by

$$x^{(k+1)} = \phi(x^{(k)}), \quad k = 0,1,2,\ldots. \tag{3.17}$$

Theorem 3.1 gives sufficient conditions for the convergence of the Newton sequence starting from $x^{(0)}$ to a solution x^* of equation (1.2). The Krawczyk operator K defined by (3.11) stems from an adaptation of the Newton operator (3.16) and the iteration process (3.17) to interval computation [8]. Under the hypotheses of Theorem 3.2, the operator ψ defined by

$$\psi(x) = x - Y\,f(x) \tag{3.18}$$

will have a fixed point $x^* \in K(X)$, which is also a solution of equation (1.2) by the invertibility of Y [12].

4. Moore's theorem in R^n_∞. Because of the different character of Theorems 3.1 and 3.2, it will be necessary to make some special assumptions in order to compare them. As the metric topology for intervals [11, pp. 15-24] is closely related to the norm

$$(4.1) \qquad \|x\| = \max_{(i)} \{|x_i|\}$$

for R^n_∞, the method of comparison will be to reformulate Moore's theorem as a (less general) metric theorem, which will then be compared to the Kantorovich theorem in R^n_∞. The following concepts will be needed.

The closed ball $\bar{U}(y,\rho)$ in R^n_∞ with center y and radius ρ is a special type of interval, namely

$$(4.2) \qquad \bar{U}(y,\rho) = \{x : \|x-y\| \leq \rho\} = [y-\rho e, y+\rho e] ,$$

where $e = (1,1,\ldots,1)$. In particular, the *closed unit ball* $\bar{U}(0,1)$ in R^n_∞ is simply the interval $[-e,e]$. The *magnitude* of the scalar interval $[\alpha,\beta]$ in R is defined to be

$$(4.3) \qquad |[\alpha,\beta]| = \max\{|\alpha|,|\beta|\} .$$

Similarly, for an interval $X = [a,b]$ in R^n,

$$(4.4) \qquad |X| = \max_{(i)} \{\max(|a_i|,|b_i|)\} = \max\{\|a\|,\|b\|\} = \max_{x \in X} \|x\|$$

in terms of the norm (4.1). The *width* of X is

$$(4.5) \qquad w(X) = \max_{(i)} \{(b_i - a_i)\} = \|b-a\| ,$$

and the *midpoint* of X is $m(X) = \frac{1}{2}(a+b)$. The matrix norm corresponding to (4.1) for matrices $A = (a_{ij})$ is

$$(4.6) \qquad \|A\| = \max_{(i)} \sum_{j=1}^{n} |a_{ij}| .$$

It follows from this and (4.4) that

$$(4.7) \qquad \|A\| = |A[-e,e]| .$$

For a matrix $M = ([\alpha_{ij}, \beta_{ij}])$ with interval components, one has

$$(4.8) \qquad |M| = \max_{(i)} \sum_{j=1}^{n} |[\alpha_{ij},\beta_{ij}]| = \max_{A \in M} \{\|A\|\} = |M[-e,e]| .$$

In Theorem 3.2, suppose that $B \geq \|Y\|$, $\eta \geq \|y-w\|$, where $w = y - Y f(y)$, and

$$(4.9) \qquad X_\rho = [y - \rho e, y + \rho e] = \bar{U}(y,\rho) .$$

The Krawczyk transformation of X is thus

$$(4.10) \qquad K(X_\rho) = w + \rho\{I - YF'(X_\rho)\}[-e,e] .$$

Also, suppose that

$$(4.11) \qquad \omega(\rho) \geq |Y^{-1} - F'(X_\rho)| .$$

Lemma 4.1. If $B\omega(\rho) < 1$, then $K(X_\rho) \subset X_\rho$ for

$$\rho \geq \frac{\eta}{1 - B\omega(\rho)} . \tag{4.12}$$

Proof: As I, Y are real matrices, one may write

$$\{I-YF'(X)\} = Y\{Y^{-1}-F'(X)\} . \tag{4.13}$$

For $v \in \rho\{I-YF'(X)\}[-e,e]$, $(w+v) \in K(X_\rho)$, and

$$\|y - (w+v)\| \leq \|y-w\| + \|v\| \leq \eta + B\rho\omega(\rho), \tag{4.14}$$

so that $\|y - (w+v)\| \leq \rho$ if (4.12) holds, and thus $(w+v) \in X_\rho$. QED.

An interval of the form $[-a,a]$ is said to be *symmetric*. As a linear transformation of a symmetric interval is symmetric, one has

$$\{I - YF'(X_\rho)\}[-e,e] = [-c,c], \tag{4.15}$$

and thus $K(X_\rho) = [w-\rho c, w+\rho c]$. The next step in the comparison of Theorems 3.1 and 3.2 will be to use the Lipschitz continuity of f' to obtain a scalar majorant function for $\omega(\rho)$. If f' is Lipschitz continuous on an interval X (with Lipschitz constant κ), then each component f'_{ij} of f' is also Lipschitz continuous on X, that is

$$|f'_{ij}(u) - f'_{ij}(v)| \leq \lambda_{ij}\|u-v\| , \quad u,v \in X . \tag{4.16}$$

Define

$$\lambda = \max_{(i)} \sum_{j=1}^{n} \lambda_{ij} . \tag{4.17}$$

It follows that λ will be a Lipschitz constant for f' on X, so that one may assume $\kappa \leq \lambda$.

Lemma 4.2. If $y \in X$, then

$$|f'(y) - F'(X)| \leq \lambda|y-X|. \tag{4.18}$$

Proof: By (3.10), (4.4), and (4.16),

$$|f'_{ij}(y) - F'_{ij}(X)| \leq \lambda_{ij} \max_{x \in X} \{\|y-x\|\} = \lambda_{ij}|y-X|, \tag{4.19}$$

$i,j = 1,2,\ldots,n$. Inequality (4.18) now follows from (4.8), (4.16), and (4.17). QED.

The choice $Y = [f'(y)]^{-1}$ now gives a metric theorem which allows direct comparison of Theorems 3.1 and 3.2.

Theorem 4.1. If

$$h = B\lambda\eta \leq \frac{1}{4} , \tag{4.20}$$

then $K(X_\rho) \subset X_\rho$ for ρ such that

$$\frac{1 - \sqrt{1-4h}}{2h}\eta \leq \rho \leq \frac{1 + \sqrt{1-4h}}{2h}\eta \tag{4.21}$$

and $X_\rho \subset X$.

Proof: As $y = m(X_\rho)$ is the midpoint of X_ρ, $|y-X_\rho| = \frac{1}{2}w(X_\rho) = \rho$, and one may take

$$\omega(\rho) = \lambda\rho \tag{4.22}$$

by (4.11) and (4.18). Thus, if $h \leq \frac{1}{4}$, then inequality (4.12) may be solved to give (4.21), and the conclusion follows from Lemma 4.1. QED.

In order to use Theorem 4.1 to compare the theorems of Kantorovich and Moore, it is natural to take $y = x^{(0)}$, $Y = Y_0$, where

$$Y_0 = [f'(x^{(0)})]^{-1}, \tag{4.23}$$

so that $w = x^{(1)}$, $B = B_0$, $\eta = \eta_0$. Thus, for

$$X^{(0)} = [x^{(0)} - 2\eta_0 e, x^{(0)} + 2\eta_0 e] = \Omega, \tag{4.24}$$

one has

$$K(X^{(0)}) = x^{(1)} + 2\eta_0\{I-Y_0F'(X^{(0)})\}[-e,e]. \tag{4.25}$$

Corollary 4.1. Under the hypotheses of Theorem 3.1, if

$$h_0 \leq \frac{\kappa}{4\lambda}, \tag{4.26}$$

then $K(X^{(0)}) \subset X^{(0)}$.

Proof: This follows immediately from Theorem 4.1, as $h_0 = (\lambda/\kappa)h$. QED.

<u>Remark 4.1.</u> As (4.26) requires that $h_0 \leq \frac{1}{4}$ even if $\kappa = \lambda$, Corollary 4.1 provides a comparison of Theorem 3.1 with Theorem 3.2 which is unfavorable to the latter. Furthermore, this cannot be improved in general, as the following example shows. In R, take $f(x) = x^2 - \varepsilon^2$ for $0 \leq \varepsilon \leq 1$ and $x^{(0)} = 1$. This gives

$$x^{(1)} = \frac{1}{2}(1 + \varepsilon^2), \quad \eta_0 = \frac{1}{2}(1 - \varepsilon^2), \tag{4.27}$$

from which $X^{(0)} = [\varepsilon^2, 2 - \varepsilon^2]$. As $B_0 = \frac{1}{2}$ and $\kappa = \lambda = 2$, it follows that

$$h_0 = h = \frac{1}{2}(1 - \varepsilon^2), \tag{4.28}$$

and Theorem 3.1 guarantees the existence of the solution $x^* = \varepsilon$ of $f(x) = 0$ in $X^{(0)}$ for $0 \leq \varepsilon \leq 1$. On the other hand, direct computation with (4.25) yields

$$K(X^{(0)}) = [-\frac{1}{2} + \frac{5}{2}\varepsilon^2 - \varepsilon^4, \frac{3}{2} - \frac{3}{2}\varepsilon^2 + \varepsilon^4], \tag{4.29}$$

and it is easy to verify that $K(X^{(0)}) \subset X^{(0)}$ if and only if $\frac{1}{2} \leq \varepsilon^2 \leq 1$, that is, $0 \leq h_0 \leq \frac{1}{4}$.

A result more favorable to Moore's theorem may be obtained by making different choices of y, Y, and X than those above. If the hypotheses of Theorem 3.1 are satisfied, then the Newton sequence $\{x^{(k)}\}$ generated by (3.17) is well defined, as are the sequences of real numbers $\{\eta_k\}$, $\{B_k\}$, $\{h_k\}$ satisfying the relationships

$$(4.30)\quad \begin{cases} \eta_{k+1} = \frac{1}{2}\,\frac{h_k\eta_k}{1-h_k} \geq \|x^{(k+2)} - x^{(k+1)}\|, \\ B_{k+1} = \frac{B_k}{1-h_k} \geq \|[f'(x^{(k+1)})]^{-1}\|, \\ h_{k+1} = \frac{1}{2}\left(\frac{h_k}{1-h_k}\right)^2 = B_{k+1}\kappa\,\eta_{k+1}\,, \end{cases}$$

$k = 0,1,2,\ldots$ [6; 15, pp. 135-138]. It is easy to verify that $\eta_k \leq 2^{-k}\eta_0$, and thus

$$(4.31)\qquad X^{(k)} = [x^{(k)} - 2\eta_k e\,,\ x^{(k)} + 2\eta_k e] \subset X^{(0)}.$$

Theorem 4.2. If the hypotheses of Theorem 3.1 hold with

$$(4.32)\qquad h_0 < \frac{1}{2}\,,$$

then there exists a positive integer k such that for $y = x^{(k)}$, $Y = Y_k = [f'(x^{(k)})]^{-1}$, one has $K(X^{(k)}) \subset X^{(k)}$.

Proof: If (4.32) holds, then the numbers h_k decrease rapidly with k. In fact, the recurrence relations (4.30) can be solved [3] to give

$$(4.33)\qquad h_k = \frac{2\theta_0^{2^k}}{\left[1+\theta_0^{2^k}\right]^2}\,,\quad k = 0,1,2,\ldots\,,$$

where

$$(4.34)\qquad \theta_0 = \frac{1-\sqrt{1-2h_0}}{1+\sqrt{1-2h_0}}\,,$$

and $\theta_0 < 1$ if (4.32) holds. Thus, a positive integer k will exist such that

$$(4.35)\qquad h_k \leq \frac{\kappa}{4\lambda}$$

and the conclusion will follow from Theorem 4.1, as $h = (\lambda/\kappa)h_k \leq \frac{1}{4}$. QED.

Remark 4.2. The condition (4.32) implies quadratic convergence of the Newton sequence $\{x^{(k)}\}$ to x^*; furthermore, x^* will be unique in $X^{(0)}$ and a *simple* zero of f in the sense that $[f'(x^*)]^{-1}$ will exist [16]. The case $h_0 = \frac{1}{2}$ is a borderline situation, which corresponds to linear convergence of the Newton sequence with ratio $\frac{1}{2}$ [6]. Consequently, if rapid convergence of the Newton sequence to the approximate solution y has been observed computationally, then it is likely that the value of h_0 corresponding to $x^{(0)} = y$ satisfies (4.26) immediately so that either Theorem 3.1 or 3.2 is applicable.

5. Precision and sensitivity to simple zeros. On the assumption that h_0 satisfies (4.26), it follows from Thereom 3.1 that for $y = x^{(0)}$,

$$\|y - x^*\| \le \frac{1 - \sqrt{1-2h_0}}{h_0}\, \eta_0 = \rho_0 , \tag{5.1}$$

and, similarly, Theorem 4.1 gives, by (4.20),

$$\|y - x^*\| \le \frac{1 - \sqrt{1-4(\lambda/\kappa)h_0}}{2(\lambda/\kappa)h_0}\, \eta_0 = \rho . \tag{5.2}$$

If $0 < h_0 \le \kappa/4\lambda$, then it is evident that

$$\rho_0 < \rho , \tag{5.3}$$

so that the Kantorovich theorem is of greater precision than the metric version of Moore's theorem. However,

$$\lim_{h_0 \to 0} \frac{\rho_0}{\rho} = 1 , \tag{5.4}$$

so that the difference may be inconsequential for h_0 sufficiently small.

To compare sensitivity, suppose that x^* is a simple zero of f, and

$$B^* \ge \|[f'(x^*)]^{-1}\|. \tag{5.5}$$

Given Lipschitz constants κ, λ in a sufficiently large region Ω containing x^*, one may take

$$h_0 = \frac{1 - \frac{1}{2} B^* \kappa\|x^{(0)}-x^*\|}{(1-B^* \kappa\|x^{(0)}-x^*\|)^2}\, B^* \kappa\|x^{(0)}-x^*\| \tag{5.6}$$

in Theorem 3.1, provided that $\|x^{(0)}-x^*\| < 1/B^*\kappa$ [16]. From (5.6),

$$\|x^{(0)} - x^*\| \le \frac{1}{B^*\kappa}\left[1 - \frac{\sqrt{1+2h_0}}{1+2h_0}\right] = \sigma(h_0) , \tag{5.7}$$

which may be used as a measure of sensitivity. From Theorem 3.1, x^* will be detected if

$$\|x^{(0)} - x^*\| \le \sigma(\tfrac{1}{2}) , \tag{5.8}$$

that is, if $x^{(0)} \in \bar{U}(x^*, \sigma(\frac{1}{2}))$, while Corollary 4.1 requires that

$$\|x^{(0)} - x^*\| \le \sigma\left(\frac{\kappa}{4\lambda}\right) . \tag{5.9}$$

Thus, as

$$\frac{\sigma(\frac{1}{4})}{\sigma(\frac{1}{2})} = \frac{2}{3}\left(\frac{3-\sqrt{6}}{2-\sqrt{2}}\right) = 0.6265\ldots , \tag{5.10}$$

the best radius of detection of x^* that can be guaranteed for the metric version of Moore's theorem is about 62.5% of the corresponding value for the Kantorovich theorem. The example of Remark 4.1 can also be used to show that this cannot be improved in general. As in Theorem 4.2, however, if $\|x^{(0)} - x^*\| < \sigma(\frac{1}{2})$, then $h_0 < \frac{1}{2}$, and there will exist a positive integer k such that

$$\|x^{(k)} - x^*\| \leq \sigma\left(\frac{\kappa}{4\lambda} \right) \tag{5.11}$$

for $x^{(k)}$ belonging to the Newton sequence $\{x^{(k)}\}$, and Moore's theorem with $y = x^{(k)}$, $Y = Y_k = [f'(x^{(k)})]^{-1}$, $X^{(k)} = [x^{(k)} - 2\eta_k e, x^{(k)} + 2\eta_k e] = X$ will detect x^*. The number of iterations required to obtain $\|x^{(k)} - x^*\| \leq \sigma(\frac{1}{4})$ is shown in Table 5.1 for various values of h_0.

h_0	k
0.41421	1
0.47648	2
0.49397	3
0.49848	4
0.49962	5
0.49990	6
0.49997	7
0.49999	8

Table 5.1.

Iterations Required to Ensure $h_k \leq \frac{1}{4}$.

6. Computational complexity. The computer program [9] for the implementation of the Kantorovich theorem has been adapted to apply Moore's theorem for the choices of y, Y, and X indicated for Corollary 4.1, and, as an additional option, for the choice

$$(6.1) \qquad Y = [m(F'(X))]^{-1}$$

recommended by Moore and Jones [13]. This program provides an experimental as well as a theoretical basis for the comparison of the computational complexity of Theorem 3.1 with Theorem 3.2.

To make effective use of either theorem, software is needed for automatic differentiation [7] and implementation of interval computation for arithmetic operations and functions ordinarily encountered in FORTRAN expressions [17] . In addition, the program for the Kantorovich theorem requires software for interval matrix inversion [5; 11, pp. 32-39] as part of the painstaking calculations required to guarantee that upper bounds B_0, κ, η_0 are obtained for the quantities indicated in (3.1), (3.2), and (3.4) as results of inexact computations with machine numbers.

As the endpoints of intervals employed in actual calculations must be machine numbers, directed rounding is used after each operation to preserve the inclusion relationship. For this and other reasons [11], instead of the exact interval extension G of a continuous function g , a *computed extension* $\bar{G}$ is obtained, which has the property that $\bar{G}(X) \supset G(X)$ for all X . Thus, $|\bar{G}(X)| \geq |G(X)|$ and $\bar{G}(X) \subset X$ implies $G(X) \subset X$, so that rigorous conclusions can be drawn on the basis of interval calculations with computed extensions. In what follows, it will be convenient to identify a machine (or real) number z with the *degenerate interval* $z = [z,z]$.

The calculation of B_0, η_0 for the Kantorovich theorem are fairly straightforward applications of interval analysis. The Lipschitz constant κ, however, will be obtained from a bound for the second derivative

$$(6.2) \qquad f''(x) = \left(\frac{\partial^2 f_i(x)}{\partial x_j \, \partial x_k} \right) ,$$

which will be called the *Hessian* of f . For a real bilinear operator $Q = (q_{ijk})$ in R_∞^n ,

$$(6.3) \qquad \|Q\| = \max_{\|x\|=1} \|Qx\| \leq \max_{(i)} \sum_{j=1}^{n} \sum_{k=1}^{n} |q_{ijk}| = \mu(Q) ,$$

with a similar expression for $|Q|$ if the elements of Q are intervals. Thus, a machine number κ such that

(6.4) $$\kappa \geq \mu(\bar{F}''(X)) = \max_{(i)} \sum_{j=1}^{n} \sum_{k=1}^{n} |\bar{F}''_{ijk}(X)|,$$

where $\bar{F}''$ is the computed extension of f'', will be a Lipschitz constant for f' on X, as

(6.5) $$\kappa \geq \max_{x\epsilon X} \|f''(x)\|.$$

The essential interval operations to implement the Kantorovich theorem will now be listed, it being assumed that $x^{(0)} = [x^{(0)}, x^{(0)}]$ is an exact machine vector:

K1. $\bar{F}(x^{(0)})$ is evaluated.

K2. The interval Jacobian $\bar{F}'(x^{(0)})$ is calculated.

K3. $\bar{Y}_0 \supset [F'(x^{(0)})]^{-1}$ is obtained by interval matrix inversion of $\bar{F}'(x^{(0)})$.

K4. $B_0 \geq |\bar{Y}_0| \geq \|[f'(x_0)]^{-1}\|$ is obtained by interval arithmetic, using (4.8).

K5. $\bar{W} = x^{(0)} - \bar{Y}_0 \bar{F}(x^{(0)})$ is computed using interval arithmetic, so that $x^{(1)} \epsilon \bar{W}$ for the exact Newton point.

K6. $\eta_0 \geq |x^{(0)} - W| \geq \|x^{(1)} - x^{(0)}\|$ is obtained from (4.4) by interval arithmetic.

K7. $\bar{X} \supset [x^{(0)} - 2\eta_0 e, x^{(0)} + 2\eta_0 e]$ is constructed, using interval arithmetic.

K8. $\bar{F}'(\bar{X})$ is evaluated to obtain

K9. $\bar{F}''(\bar{X})$ by differentiation.

K10. $\kappa \geq \mu(\bar{F}''(\bar{X})) \geq \max_{x\epsilon\bar{X}} \|f''(x)\|$ is obtained from (6.4) by interval arithmetic.

K11. $h_0 \geq |B_0 \kappa \eta_0|$ is calculated, using interval arithmetic.

After the machine number h_0 has been obtained in this way, it is checked versus the machine number $\frac{1}{2}$ to see if (3.5) holds to complete the automation of Theorem 3.1.

In the case of the Moore theorem, interval matrix inversion is avoided by evaluating $f'(x^{(0)})$ approximately as a real matrix, say in double precision, and then inverting the result carefully to obtain a real matrix Y which is the inverse of <u>some</u> matrix. (A failure of matrix inversion here or in K3 results in an exit from the program with an appropriate error indication.) The interval computations required for Moore's theorem with $y = x^{(0)}$ are:

M1. $\bar{F}(y)$ is evaluated.

M2. $\bar{W} = y - Y\bar{F}(y)$ is computed using interval arithmetic; one has $w \epsilon \bar{W}$.

M3. $\eta \geq |\bar{W}|$ is obtained from (4.4) by interval arithmetic.

M4. $\bar{X} \supset [y - 2\eta e,\ y + 2\eta e]$ is constructed by interval arithmetic .

M5. $\bar{F}'(\bar{X})$ is evaluated.

M6. $\bar{K}(\bar{X}) \supset \bar{W} + 2\eta\{I - Y\bar{F}'(\bar{X})\}[-e,e]$ is constructed by interval arithmetic.

These calculations result in the intervals $\bar{X} = [a,b]$, $\bar{K}(\bar{X}) = [c,d]$, where the components of the vectors a,b,c,d are all machine numbers. The verification of $\bar{K}(\bar{X}) \subset \bar{X}$ thus depends on checking the 2n inequalities

(6.6) $$a_i \le c_i,\quad d_i \le b_i\ ,\quad i = 1,2,\ldots,n,$$

between machine numbers.

Comparison of the lists of interval operations for the two theorems reveals that M1 = K1, M2 $\le$ K5 (Y is a real matrix, so obtaining $Y\bar{F}'(y)$ generally requires fewer operations than for $\bar{Y}_0\bar{F}'(x^{(0)})$), M3 = K6, M4 = K7, M5 = K8. This commonality of subroutines made adaption of the original program to the new theorem very easy. This leaves K2, K3, K4, K10, K11 for implementation of Theorem 3.1 as against M6 and the inversion of a real matrix for automation of Theorem 3.2. In particular, K3 (interval matrix inversion) is tedious, and K9 (Hessian evaluation) requires $\frac{1}{2}n^2(n+1)$ differentiations and interval evaluations, while M6 is a simple interval matrix-vector multiplication.

It follows that the application of Theorem 3.2 is less complex by far than the use of Theorem 3.1. At least for good approximations $y = x^{(0)}$ to x^*, there is no significant difference in precision or sensitivity to offset this advantage of Moore's theorem.

A comment is in order on the choice (6.1) for Y . For $X = X_\rho$,

(6.7) $$|m(F'(X_\rho)) - F'(X_\rho)| = \frac{1}{2}w(F'(X_\rho))\ ,$$

and from (4.16) and (4.17),

(6.8) $$w(F'(X_\rho)) \le \lambda w(X_\rho) = 2\lambda\rho\ .$$

Thus, one may take $\omega(\rho) = \lambda\rho$ in Lemma 4.1, which gives Theorem 4.1 with

(6.9) $$B \ge \|[m(F'(X_\rho))]^{-1}\|\ ,\quad \eta \ge \|w-y\| = \|Yf(y)\|\ .$$

Here, however, ρ is chosen first, and (4.21) must hold if $h = B\lambda\eta \le \frac{1}{4}$. Computationally, this method differs from the above in that M_3 and M_4 are eliminated, M5 follows the choice of X_ρ , $m(\bar{F}'(X_\rho))$ is then evaluated and inverted to obtain Y and then M1, M2, and M6 with $2\eta = \rho$ complete the computation. There is thus no essential difference in complexity.

7. A numerical example. An actual comparison of computational efficiency was obtained by applying the computer program [9], modified to include the implementation of Moore's theorem, to the quadratic system

$$(7.1) \qquad x_i - \alpha x_i \sum_{j=1}^{9} a_{ij} x_j - 1 = 0 \ , \quad i = 1,2,\ldots,9,$$

with

$$(7.2) \qquad a_{ij} = \frac{3}{4} \; \frac{t_i(1-t_j^2)^2 w_j}{t_i + t_j} \ ,$$

where t_i, w_i, $i = 1,2,\ldots,9$, are respectively the nodes and weights of the Gaussian integration rule of order 17 on the interval $0 \le t \le 1$ [10, p. 288]. The system (7.1) was constructed as a discrete approximation to the nonlinear integral equation

$$(7.3) \qquad x(s) = 1 + \frac{3\alpha}{4} s \, x(s) \int_0^1 \frac{(1-t^2)^2}{s+t} x(t)dt, \qquad 0 \le s \le 1 \ ,$$

which is a case of the H-equation of Chandrasekhar [1, p. 105]. The value of α considered was $\alpha = 0.7$, and the initial approximation $y = x^{(0)} = e$ gave

$$(7.4) \qquad \bar{X} = [0.3405052e, 1.6594949e].$$

(Directed rounding is used in the conversion of intervals from binary to decimal; hence, the decimal interval $\bar{X}$ given by (7.4) contains the binary interval stored in the computer.)

The program for the Kantorovich theorem computed

$$(7.5) \qquad h_0 = 0.700800 > \frac{1}{2} \ ,$$

and thus failed to detect the existence of a solution $x^* = (x_1^*, x_2^*, \ldots, x_9^*)$ of (7.1) in $\bar{X}$. A total of 21.4652 UNIVAC 1110 time units were required, of which 4.7436 were expended for interval matrix inversion (K3 and K4), and 12.1394 were required for the evaluation of the Hessian (K9 and K10).

The program for the Moore theorem, on the other hand, required only a total of 5.4120 time units, and gave $\bar{K}(\bar{X}) \subset \bar{X}$, thus guaranteeing the existence of a solution $x^* \in \bar{K}(\bar{X})$. The components of $\bar{K}(\bar{X}) = [c,d]$ are given in Table 7.1.

In this simple example, the value of h_0 has not been overestimated badly, as the Hessian is constant, and all its nonzero elements have the same sign. For bilinear operators Q of this type, $\mu(Q) = \|Q\|$. Thus, the program based on Moore's theorem provides more information in this case, as well as showing the expected decrease in execution time. For systems with nonconstant Hessians, the advantage in speed of Theorem 3.2 should be even greater than the 4 to 1 advantage observed here. In addition, the flexibility of being able to use intervals other

i	c_i	d_i
1	1.0042228	1.0606792
2	1.0135268	1.1943671
3	1.0223478	1.3211143
4	1.0293528	1.4217681
5	1.0344997	1.4957230
6	1.0381216	1.5477668
7	1.0405689	1.5829316
8	1.0421079	1.6050443
9	1.0429109	1.6165838

Table 7.1. The Interval $\bar{K}(\bar{X}) = [c,d]$.

than balls and the componentwise error bounds furnished by intervals may be of significant advantage in practical computation.

The Author wishes to acknowledge the assistance of Julia H. Gray, S. T. Jones, and C. T. Kelley with this example, which arose in an investigation completely unrelated to the subject of this paper.

REFERENCES

1. S. Chandrasekhar, Radiative Transfer, Dover, New York, 1960.

2. G. E. Forsythe and C. B. Moler, Computer Solution of Linear Algebraic Systems, Prentice-Hall, Englewood Cliffs, N.J., 1967.

3. W. B. Gragg and R. A. Tapia, Optimal error bounds for the Newton-Kantorovich theorem, SIAM J. Numer. Anal. 11 (1974), 10-13.

4. Julia H. Gray and L. B. Rall, NEWTON: A general purpose program for solving nonlinear systems, Proceedings of the 1967 Army Numerical Analysis Conference, pp. 11-59. U. S. Army Research Office, Durham, N.C., 1967.

5. E. Hansen, Interval arithmetic in matrix computations, Part I, SIAM J. Numer. Anal. 2 (1965), 308-320.

6. L. V. Kantorovich, Functional analysis and applied mathematics, Uspehi Mat. Nauk 3 (1948), 89-185 (Russian). Tr. by C. D. Benster, Natl. Bur. Std. Rept. No. 1509, Washington, D. C., 1952.

7. G. Kedem, Automatic differentiation of computer programs, Technical Summary Report No. 1697, Mathematics Research Center, University of Wisconsin-Madison, 1976.

8. R. Krawczyk, Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken, Computing 4 (1969), 187-201.

9. Dennis Kuba and L. B. Rall, A UNIVAC 1110 program for obtaining rigorous error estimates for approximate solutions of systems of equations, Technical Summary Report No. 1168, Mathematics Research Center, University of Wisconsin-Madison, 1972.

10. W. E. Milne, Numerical Calculus, Princeton University Press, Princeton, N.J., 1949.

11. R. E. Moore, Interval Analysis, Prentice-Hall, Englewood Cliffs, N.J., 1966.

12. R. E. Moore, A test for existence of solutions to nonlinear systems, SIAM J. Numer. Anal. 14 (1977), 611-615.

13. R. E. Moore and S. T. Jones, Safe starting regions for iterative methods, SIAM J. Numer. Anal. 14 (1977), 1051-1065.

14. J. M. Ortega, The Newton-Kantorovich theorem, Amer. Math. Monthly 75 (1968), 658-660.

15. L. B. Rall, Computational Solution of Nonlinear Operator Equations, John Wiley & Sons, New York, 1969.

16. L. B. Rall, A note on the convergence of Newton's method, SIAM J. Numer. Anal. 11 (1974), 34-36.

17. J. M. Yohe, Implementing nonstandard arithmetics, SIAM Rev. 21 (1979), 34-56.

Subgrid Resolution of Fluid Discontinuities

James Glimm and Dan Marchesin
The Rockefeller University
New York, New York 10021

ABSTRACT. A simple modification in the sampling method for one dimensional fluid calculations allows tracking of discontinuity fronts with accuracy in the range .05 Δx to .5 Δx in representative problems in one space dimension.

1. INTRODUCTION.

The sampling method [1,3] provides the most accurate calculation of one dimensional fluid problems with discontinuity fronts [10]. In this method, the resolution is 100%, in the sense that there is no numerical viscosity or diffusion to blur the discontinuity front. Since chemical reaction rates are in some cases governed by diffusion, cf[5], the ability to compute with small or zero diffusion is an important practical advantage of this method.

The accuracy of the sampling method is also good; in typical runs, the discontinuity id located within approximately Δx of its exact value. The purpose of this paper is to show how a simple tracking procedure can improve on this accuracy by a factor of 4 to 10 and locate the discontinuity with accuracy in the range .05 Δx to .5 Δx. We believe this accuracy could be improved upon, and as a preliminary step to doing this, we identify some of the remaining sources of error.

2. THE METHOD OF TRACKING DISCONTINUITIES.

In the sampling method, the solution at time $t = \ell \Delta t$ is taken to be piecewise constant between mesh points. An exact solution can then be given for

$$\ell \Delta t \le t < (\ell+1) \Delta t \ . \tag{2.1}$$

In fact each jump discontinuity at a mesh point gives rise to a Riemann problem [4], which for an n x n system, resolves into n noninteracting waves, on the time interval (2.1). The differential equation, in the form of a nonlinear conservation law,

$$U_t + f(U)_x = 0 \tag{2.2}$$

and associated entropy condition defines uniquely the solution of these Riemann problems. By this construction, an approximate solution $U^{(\Delta x)}$ is obtained. However $U^{(\Delta x)}$ is not piecewise constant for $t = (\ell+1) \Delta t - 0$. By a sampling procedure, depending on a sequence $\{\theta_n\}$ equidistributed [6] in [0,1], we define

$$U^{(\Delta x)}((j+a) \Delta x, (\ell+1) \Delta t) = U^{(\Delta x)}((j+\theta_{\ell+1}) \Delta x, (\ell+1) \Delta t - 0) \text{ for } 0 \le a < 1 \ . \tag{2.3}$$

Then

$$U^{(\Delta x)} \Big|_{t = (\ell+1) \Delta x + 0} \tag{2.4}$$

is piecewise constant between mesh points, so the inductive construction of $U^{(\Delta x)}$ can continue.

In less formal language, $U^{(\Delta x)}$ is forced to be piecewise constant between mesh points, and the value of this constant (2.4), is determined by sampling the values

$$U^{(\Delta x)}\Big|_{t = (\ell+1)\,\Delta x - 0} \ . \tag{2.5}$$

For a more detailed account, see [1,3,6,7,8] .

The modification we propose in the sampling method is suitable for problems with a small number of strong waves moving in a background of weak waves. The method is a minimal mesh refinement, which simply adds one new mesh point for each discontinuity which is being tracked. The new mesh point, for $t = (\ell+1)\,\Delta t$, is located exactly on the discontinuity in the computed solution, (2.5). Neighboring mesh points are then deleted, to preserve a CFL stability condition, see Fig. 1. The result is that the sampling error in the discontinuity is zero. Thus the only error in the position of the discontinuity results from interaction of the discontinuity with first order errors in the computed solution in the smooth region (of small waves).

In order to implement this change easily, we use a one step scheme (i.e. the grid is not staggered). See Fig. 1.

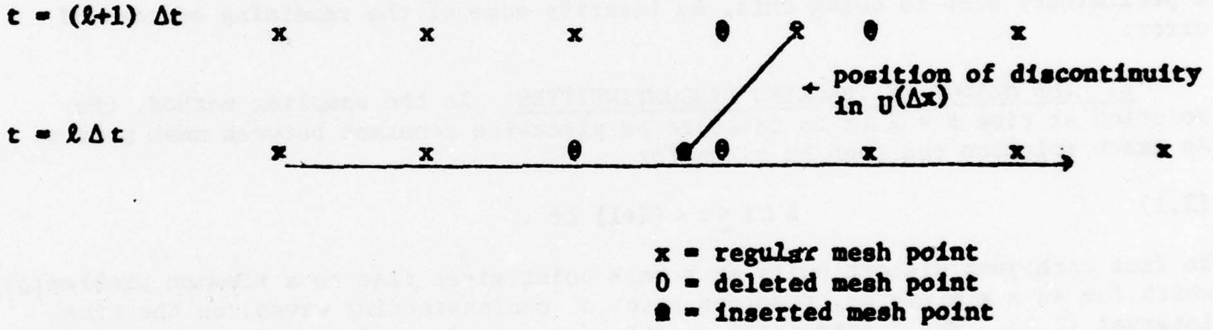


Figure 1. Mesh refinement used to track a discontinuity.

3. THE RESULTS . We use the van der Corput (vd C) equidistributed sequence [2] and the sequence n $\sqrt{2}$ (mod 1). The Cauchy data is shown in figure 2.

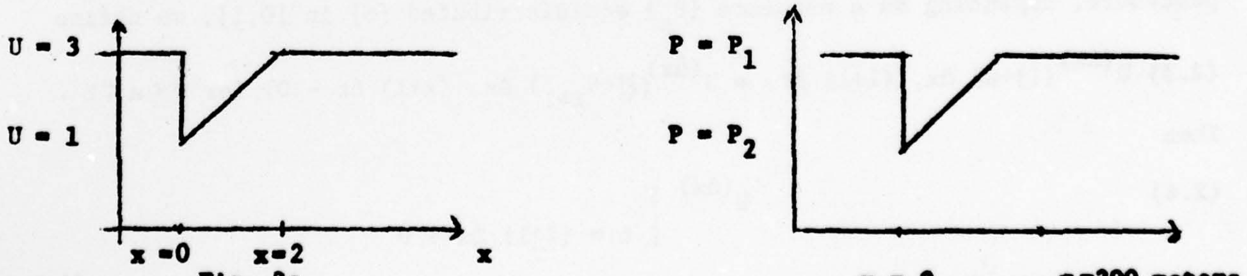


Fig. 2a

Burger's Equation

$U_t + UU_x = 0$

Fig. 2b

Polytropic gas, γ = 1.4

U = 0 . ρ = 5 kg/m^3

P_1 = 5x 10^5 n/m^2 ≈ 5 atm. P_2 = 2x 10^5 n/m^2 ≈ 2 atm.

As a measure of error δP in the shock position, we tabulate $\sup_{0 \le t \le 4} |\delta P|/\Delta x$ for Burger's equation in Table 1 below.

	Tracked Shock		Not Tracked
θ_n	vdC	$n\sqrt{2}$	vdC
$\Delta x = .1$	.29	1.1	5.7
$\Delta x = .1/2$	.28	.6	3.1
$\Delta x = .1/4$	.08	.39	5.2
$\Delta x = .1/8$	.03	1.3	6.2

Table 1. $\sup_{0 \le t \le 4} |\delta P|/\Delta x$ for Burger's equation

We compare our solution to the sampling method solution without tracking of the discontinuity because the latter gives very good solutions for this problem. For the untracked solution, the error ($\sup_t$) jumps immediately to about 2 Δx, and then increases linear in time with a speed error of approximately .75 Δx/ sec, see Fig. 3. For the tracked solution, the initial fluctuation error is almost completely eliminated, and the systematic errors are reduced, see Fig. 3 and Table 2.

	Tracked Shock		Not Tracked
θ_n	vdC	$n\sqrt{2}$	vdC
Fluctuation error	nil	nil	$\simeq \pm 2\ \Delta x$
Systematic error	.04 Δx/sec	.2 Δx/sec	.75 Δx/sec

Table 2. Size and source of errors, Burger's equation.

Note that all errors are first order; the objective of the present method is to eliminate "large" ($0(1) \times \Delta x$) first order errors,so that on practical mesh sizes, an equivalent of higher order accuracy would be obtained.

The source of the fluctuation error lies in the sampling and its elimination by tracking is obvious. One source of the systematic (shock speed) error is a stopping time effect. With or without tracking of the shock wave, the relative position of the shock wave and a neighboring rarefaction wave has fluctuations of about ± Δx. However the rarefaction wave is eliminated on its first intersection with the shock wave. Because of fluctuations, this occurs one to two x mesh units too soon. In the Cauchy data we study here, the effect is to increase the speed of the shock wave.

The excellent results of the vdC sequence are fortuitous. In fact vdC introduces a first order position error, which moves waves systematically to the right. When the rarefaction wave is to the right of the shock wave, this opposes the stopping time errors, while if the rarefaction wave were to the left, it would reinforce stopping time errors.

We do not have a theoretical explanation for the 3 1/2 - fold reduction in systematic error with $\theta_n = n\sqrt{2} \pmod 1$, but since most of the stopping time errors would probably be eliminated by a modification of the method, we do not regard the $n\sqrt{2}$ results as optimal.

We used other measures of the error δP, and obtain similar results. Roughly

$$\text{Average}_t \, |\delta P| \approx \frac{1}{2} \sup_t |\delta P|$$

for the tracked solution, indicating that fluctuations in δP are not important. Other measures of the accuracy of the solution, such as

$$\int (U_{comp} - U_{exact}) \, dx \approx .5\Delta x$$

$$\| U_{comp} - U_{exact} \|_{L_2} \approx .5\Delta x$$

show first order errors $(O(1) \times \Delta x)$.

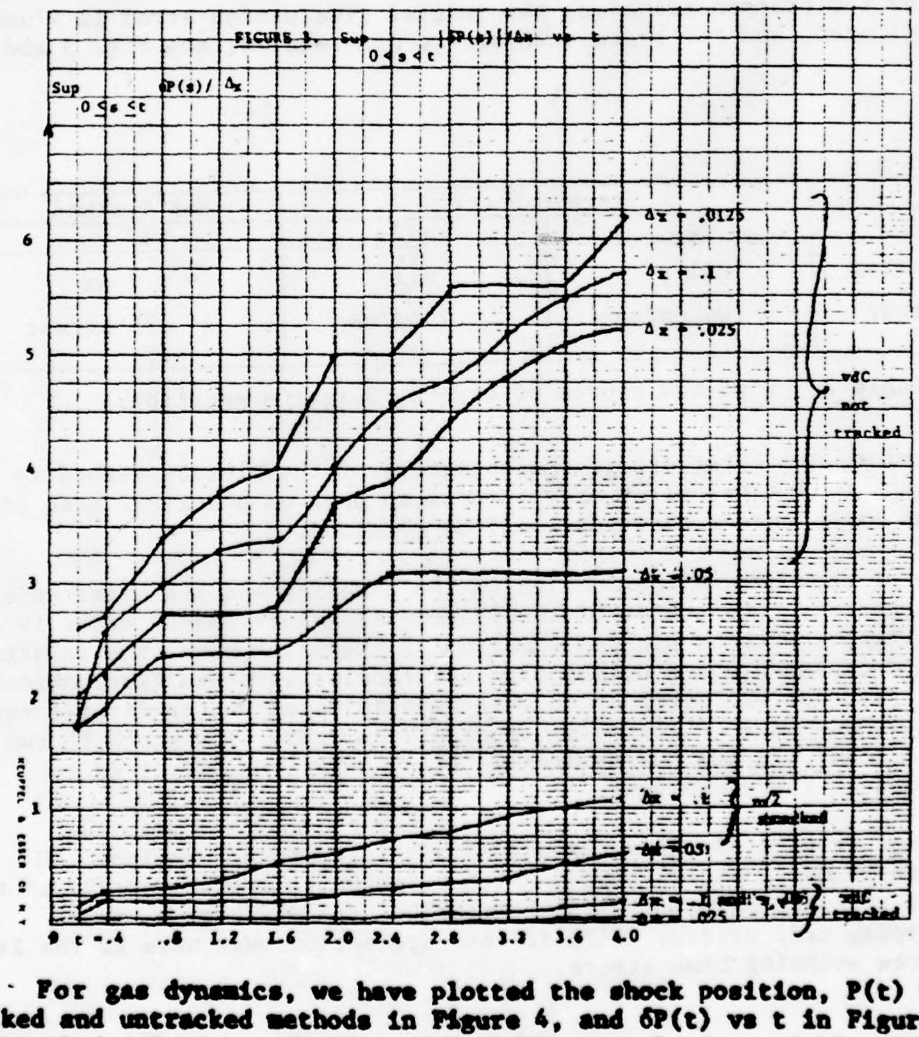


FIGURE 3. $\sup_{0 < s < t} |\delta P(s)|/\Delta x$ vs t

For gas dynamics, we have plotted the shock position, P(t) vs t for both tracked and untracked methods in Figure 4, and δP(t) vs t in Figure 5. In Fig. 4, the tracked solution coincides with the exact solution, within the accuracy of the plot.

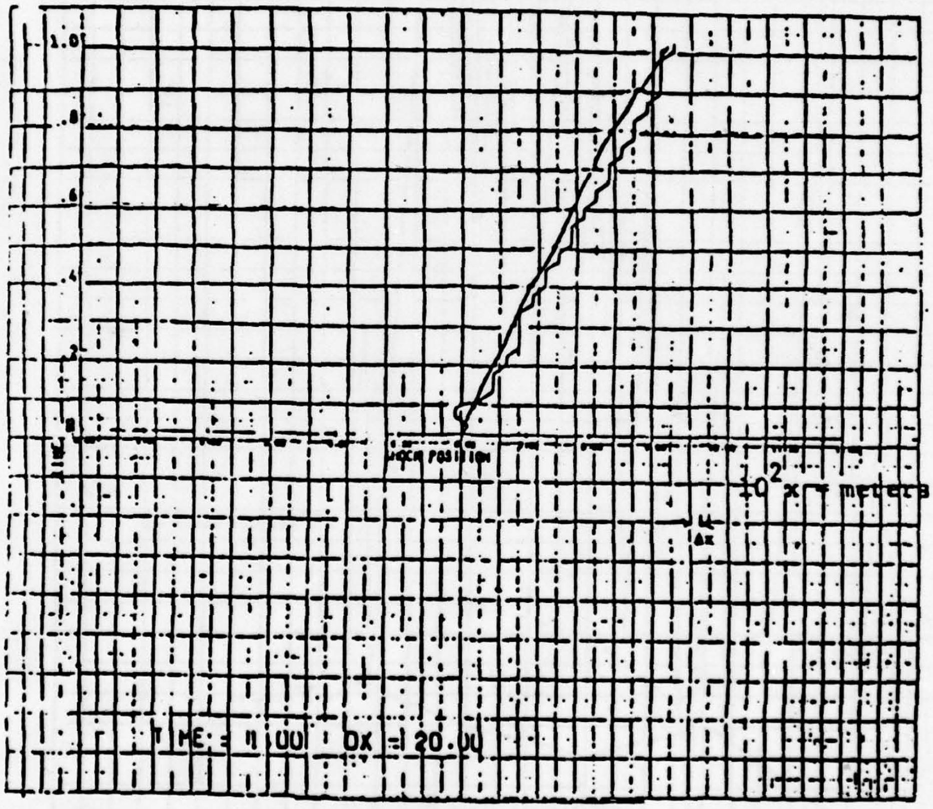


Figure 4

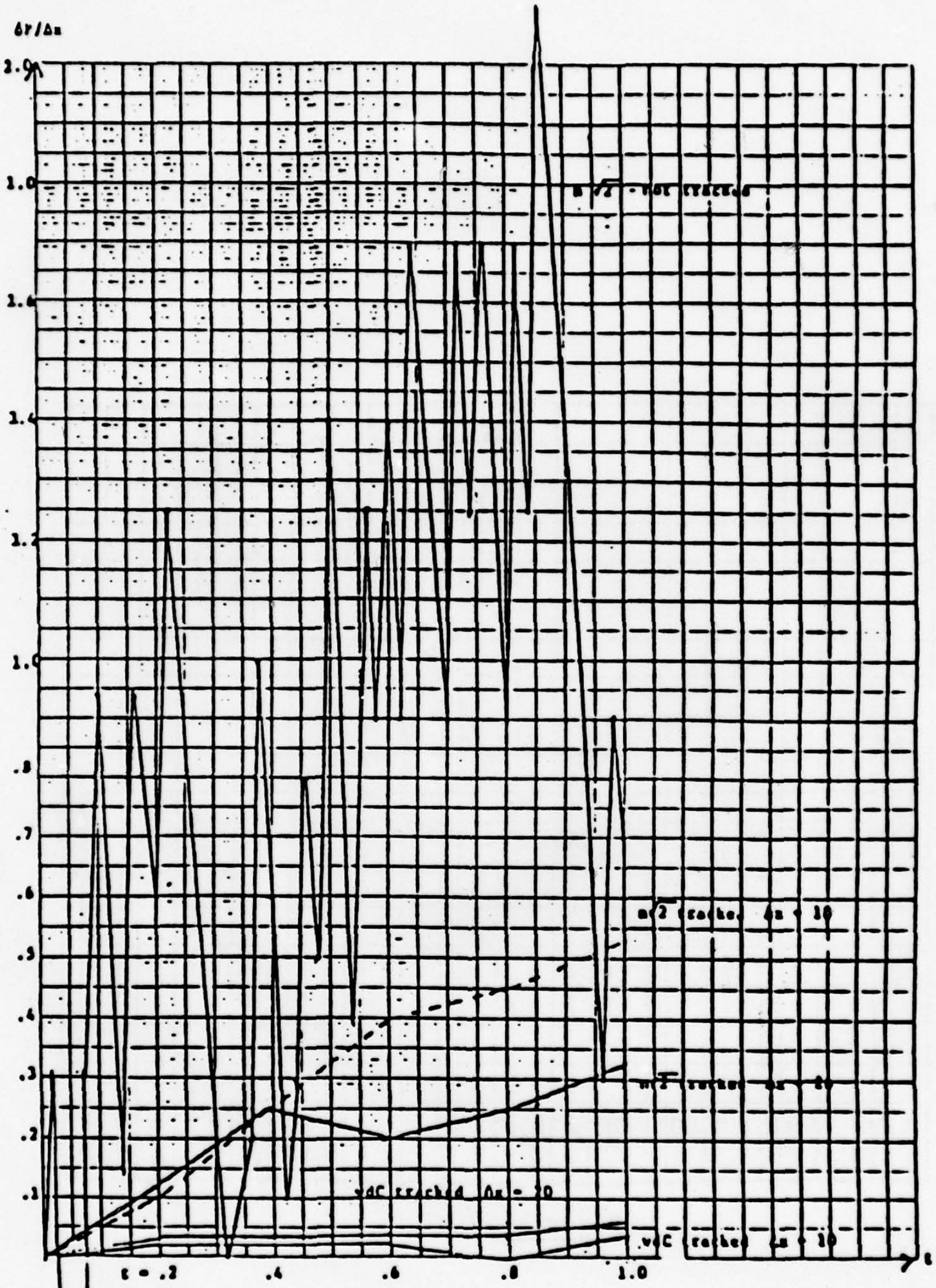


FIGURE 5

δr/Δx vs t for Gas Dynamics

x = 20 , 10 meters

Again we can separate errors into fluctuation errors and systematic errors.

	Tracked Shock		Not Tracked
θ_n	vdC	$n\sqrt{2}$	$n\sqrt{2}$
Fluctuation error	nil	nil	~ ± Δx
Systematic error	.05 Δx/sec	.4 Δx/sec	1.4 Δx/sec

Table 3. Size and source of errors, gas dynamics

The theoretical analysis is as before. In conclusion, we see that tracking of the discontinuity eliminates fluctuation errors and reduces first order systematic errors. One source (perhaps the primary source) of the remaining systematic errors has been identified.

4. DISCUSSION. We have seen that tracking of discontinuities provides a considerable increase in accuracy, but is only a minor complication to the sampling method. To remove oscillations in the smooth part of the solution, a simple effective method is to introduce a small viscosity away from all contact discontinuities and shocks [9]. Other methods can be devised which may work near discontinuities as well. It is also possible that higher order methods can be devised; however the problem of higher dimensions seems to be more important.

For two dimensional problems,we agree with Collela [2] that operator splitting of strong waves degrades the accuracy of the sampling method. An alternative, based on Huyghens principle, was proposed by the authors. This method requires at least a limited ability to track fronts in two dimensional problems, so that splitting of strong waves can be done in a geometry defined by the waves themselves, and not on a rectangular mesh. There seems to be no problem with operator splitting of weak waves [1], so that it should only be necessary to compute the new geometry near strong waves.

Bibliography

1. Chorin, A., Random Choice Solutions of Hyperbolic Systems. J. Comp. Phys. 22, p. 517 (1976).

2. Collela, P. Private communications

3. Glimm, J., Solutions in the Large for Nonlinear Hyperbolic Systems of Equations. Commun. Pure Appl. Math. 18 (1965) 697-715.

4. Lax, P. Hyperbolic Systems of Conservation Laws II, Comm. Pure Appl. Math. 10, p. 537 (1957).

5. Lin, S. University of California, Berkeley Thesis.

6. Liu, T.P., The Deterministic Version of the Glimm Scheme. Commun. Math. Phys. 57, p. 135 (1977).

7. Morawetz, C., Nonlinear Conservation Equations. Bull. Amer. Math. Soc. To Appear.

8. Nishida, T. Nonlinear Hyperbolic Equations and Related Topics in Fluid Dynamics. Universite de Paris-Sud - Departement de Mathematique.

9. Proskurowski, W., Private communication.

10. Sod, G. A Survey of Numerical Methods for Compressible Fluids. J. Comp. Phys. 27, p. 1 (1978).

DISCRETIZATIONS IN OPTIMAL CONTROL AND MESH SELECTION

G.W. Reddien
Department of Mathematics
Vanderbilt University
Nashville, Tennessee 37235

Abstract. Collocation at Gauss points is shown to be a high order accurate discretization of certain unconstrained optimal control problems. The use of adaptive mesh selection strategies is discussed.

1. Introduction. The technique of replacing a continuous optimal control problem by a discretization in order to obtain a finite dimensional approximation problem for computational purposes is an old one. Recently, high order and efficient methods for the numerical solution of two-point boundary value problems have been developed based on collocation using polynomial splines. Particularly attractive among these collocation methods are those involving collocation at Gauss points. The potential use of these schemes as discretizations in optimal control has apparently been considered only in [7] and, as we shall show, can lead to high order accurate methods.

Discretizations using polynomial splines have been considered via the Ritz-Galerkin method in [1] and [4], and the related Ritz-Trefftz principle in [5] and [6]. However, these methods generally require the use of a numerical quadrature in actual computations. The influence of such errors on high order estimates has apparently not been considered.

In section 2, the problem and method to be studied will be defined and in section 3, basic convergence results will be given for collocation at Gauss points. We consider in section 3 essentially the same class of problems that was studied in [1], including the important linear-quadratic state regulator problem.

An expansion for the error term that is valid for collocation at Gauss points when applied to two-point boundary value problems has led to the development of mesh selection strategies in [3] and [9]. In section 3, we discuss this possibility for the optimal control problems under consideration. Also, a numerical example is presented.

2. Problem and Method. Define $J(u) = \int_0^1 g(x,u,t)dt$. We consider the problem (P)

(a) $\min_u J(u)$, u in $L^2[0,1]$,

(2.0) subject to

(b) $\dot{x}(t) = f(x,u,t)$, $x(0) = x_0$, t in $[0,1]$.

We will restrict our attention here to scalar valued functions. Let $W_2^\nu[0,1]$ denote the usual Sobolev space of function [1] with ν a positive integer.

We will say that the problem (P) is in C^ν if and only if $f(x,u,t)$ and $g(x,u,t)$ are $\nu+1$ times continuously differentiable in x and u and ν times in t. The Lagrangian for (P) is

$$L(u,x,\lambda,\gamma) = J(u) + \int_0^1 (-\dot{x} + f(x,u,t))\lambda dt + (x(0)-x_0)\gamma$$

where λ is in W_2^1 and γ is scalar.

We now state three assumptions which will remain in effect throughout this paper: (A1) Problem (P) is in C^ν, $\nu \geq 1$. (A2) The Lagrangian L is extremized at the four-tuple $(u^*,x^*,\lambda^*,\gamma^*)$ satisfying the conditions

$$(2.1)\quad \begin{aligned} &(a)\quad \frac{\partial g^*}{\partial u} + (\frac{\partial f^*}{\partial u})\lambda^* = 0 \\ &(b)\quad \dot{\lambda}^* + (\frac{\partial f^*}{\partial x})\lambda^* + \frac{\partial g^*}{\partial x} = 0, \quad \lambda^*(1) = 0 \\ &(c)\quad -\dot{x}^* + f(x^*,u^*,t) = 0, \quad x^*(0) = x_0 \\ &(d)\quad \gamma^* = -\lambda^*(0) \end{aligned}$$

where the superscript * indicates the functions involved are evaluated at $u = u^*$ and $x = x^*$. Define the operator H by

$$H(u,x,\lambda) = \begin{bmatrix} g_{uu} + f_{uu}\lambda & g_{ux} + f_{ux}\lambda \\ g_{xu} + f_{xu}\lambda & g_{xx} + f_{xx}\lambda \end{bmatrix}.$$

We assume (A3) that

$$\begin{bmatrix} \delta u \\ \delta x \end{bmatrix}^T H(\tilde{u},\tilde{x},\tilde{\lambda}) \begin{bmatrix} \delta u \\ \delta x \end{bmatrix} \geq \sigma(|\delta u|^2 + |\delta x|^2)$$

where $\delta u = u-\tilde{u}$ and $\delta x = x-\tilde{x}$ with (u,x,λ) in some bounded convex neighborhood N of (u^*,x^*,λ^*) in the usual product norm on $W_2^1 \times W_2^1 \times W_2^1$ for some constant $\sigma > 0$ and uniformly in t, $0 \leq t \leq 1$. (A3) implies that the second variation of L with respect to (u,x) is strongly positive in N.

<u>Remarks</u>. Assumptions (A1), (A2), and (A3) constitute a set of local sufficiency conditions for the existence and uniqueness of a solution for Problem (P). See [1]. We will eliminate the multiplier γ from the Lagrangian by considering only variations which satisfy (2.1)(d).

The solution to (P) will be approximated using continuous polynomial splines. Let $\Delta_n : 0 = t_0 < t_1 < \ldots < t_n = 1$ be a partition of [0,1] with $|\Delta_n| \equiv \max_i(t_i - t_{i-1})$

Let $S(\Delta_n,p)$ denote the continuous polynomial splines of degree p over Δ_n (piecewise polynomials of degree $\leq p$ on each subinterval). In the ith subinterval $[t_{i-1},t_i]$ of Δ_n, let $\xi_{ij}, j = 1,\ldots,p$, denote the quadrature points for p-point Gaussian quadrature over $[t_{i-1},t_i]$ and let $w_{ij}, j = 1,\ldots,p$, denote the associated weights. With these notations, the method we study is defined as follows:

(a) $$\underset{u_n}{\text{minimize}} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} g(x_n(\xi_{ij}), u_n(\xi_{ij}), \xi_{ij})$$

(2.2)
subject to

(b) $$\dot{x}_n(\xi_{ij}) = f(x_n(\xi_{ij}), u_n(\xi_{ij}), \xi_{ij}),\ i = 1,\ldots,n;\ j = 1,\ldots,p, \quad x_n(0) = x_0,$$

where x_n and u_n are in $S(\Delta_n,p)$. In order to further simplify notation we let $\sum'_{i,j} h \equiv \sum_{i=1}^{n} \sum_{j=1}^{p} w_{ij} h(\xi_{ij})$.

The dimension of $S(\Delta_n,p)$ is $np+1$. A Lagrange basis can be constructed as follows. Find np functions $\phi_{\alpha\beta}, \alpha = 1,\ldots,n,\ \beta = 1,\ldots,p$ so that $\phi_{\alpha\beta}(\xi_{ij}) = \delta_{\alpha\beta,ij}$ where $\delta_{\alpha\beta,ij} = 1$ if $\alpha = i$ and $\beta = j$ and is zero otherwise, and $\phi_{\alpha\beta}(1) = 0$. Adjoin to this set a function ϕ_0 so that $\phi_0(1) = 1$ and $\phi_0(\xi_{ij}) = 0$. We write, for example, if $\lambda_n \in S(\Delta_n,p)$ satisfies $\lambda_n(1) = 0$, then $\lambda_n = \sum_{i=1}^{n} \sum_{j=1}^{p} \alpha_{ij}\phi_{ij}$ for some set of real coefficients α_{ij}.

3. Convergence Results. The first convergence results of this paper will follow from the necessary conditions for (2.2) which we now derive.

Theorem 3.1 Let u^*_n and x^*_n be a solution to (2.2). Then there exists a function λ^*_n in $S(\Delta_n,p)$ so that

(a) $\dot{x}^*_n(\xi_{ij}) = f(x^*_n(\xi_{ij}), u^*_n(\xi_{ij}), \xi_{ij})$

(b) $x^*_n(0) = x_0$

(3.1)

(c) $\dot{\lambda}^*_n(\xi_{ij}) = -f_x(x^*_n, u^*_n, t)|_{\xi_{ij}} \lambda^*{}_n(\xi_{ij}) - g_x(x^*_n, u^*_n, t)|_{\xi_{ij}}$

(d) $\lambda^*_n(1) = 0$

(e) $(g_u(x^*_n, u^*_n, t) + f_u(x^*_n, u^*_n, t)\ {}^*_n)|\xi_{ij} = 0,$

$i = 1,\ldots,n$

$j = 1,\ldots,p$

<u>Proof.</u> Eq. (3.1)(a) can be written first as

$$w_{ij}\phi_{ij}(\xi_{ij})\dot{x}_n(\xi_{ij}) = w_{ij}\phi_{ij}(\xi_{ij})f(x_n(\xi_{ij}),u_n(\xi_{ij}),\xi_{ij})$$

for each i,j and then using the fact that $\phi_{ij}(\xi_{\alpha\beta}) = \delta_{ij,\alpha\beta}$ as

$$\sum_{\alpha=1}^{n}\sum_{\beta=1}^{p} w_{\alpha\beta}\phi_{ij}(\xi_{\alpha\beta})\dot{x}_n(\xi_{\alpha\beta}) = \sum_{\alpha=1}^{n}\sum_{\beta=1}^{p} w_{\alpha\beta}\phi_{ij}(\xi_{\alpha\beta})f(x_n(\xi_{\alpha\beta}),u_n(\xi_{\alpha\beta}),\xi_{\alpha\beta}).$$

Thus the Lagrangian for (2.2) can be written as

$$L_n(u_n,x_n,\lambda_n) = \sum_{i,j} g(x_n,u_n,t) + \sum_{i=1}\sum_{j=1}\lambda_{ij}\sum_{\alpha,\beta}{}^{'}\phi_{ij}(-\dot{x}_n + f(x_n,u_n,t)).$$

Define $\lambda_n = \sum_{i=1}^{n}\sum_{j=1}^{p}\lambda_{ij}\phi_{ij}$. Note this implies $\lambda_n(1) = 0$ from the way that the ϕ_{ij}'s were constructed. Then L_n may be written as

$$L_n(u_n,x_n,\lambda_n) = \sum_{i,j}{}^{'}g(x_n,u_n,t) + \sum_{i,j}{}^{'}\lambda_n(-\dot{x}_n + f(x_n,u_n,t)).$$

Note that $\lambda_n\dot{x}_n$ is a piecewise polynomial of degree $2p-1$ on each subinterval of Δ_n. Thus the quadrature formula will be exact so that $-\sum_{i,j}{}^{'}\lambda_n\dot{x}_n = -\int_0^1\lambda_n\dot{x}_n dt = \int_0^1\lambda_n x_n dt + \lambda_n(0)x_n(0)$. The discrete necessary conditions are given by

$$\text{(a)}\quad \frac{\partial L_n}{\partial\lambda_n} = 0$$

$$\text{(3.2)(b)}\quad \frac{\partial L_n}{\partial x_n} = 0$$

$$\text{(c)}\quad \frac{\partial L_n}{\partial u_n} = 0.$$

From (3.2)(a) one gets back (2.3)(b). Eq(3.2)(b) gives

$$\text{(3.3)}\qquad \sum_{r,s}{}^{'}\phi_{ij}(g_x(x_n,u_n,t) + \lambda_n + f_x(x_n,u_n,t)\lambda_n) = 0$$

$$i = 1,\ldots,n$$

$$j = 1,\ldots,p$$

and Eq(3.2)(c) gives

$$(3.4) \qquad \sum_{r,s}{}' \phi_{ij}(g_u(x_n,u_n,t) + f_u(x_n,u_n,t)\lambda_n) = 0$$

$$i = 1,\ldots,n$$

$$j = 1,\ldots,p$$

Again using the fact that $\phi_{ij}(\xi_{rs}) = \delta_{ij,rs}$, Eq. (3.3) gives (3.1)(c) and Eq. (3.4) gives (3.1)(e), completing the proof.

We next want to establish the saddle point behavior of the discrete Lagrangian at a solution $(u_n^*,x_n^*,\lambda_n^*)$ in N. This result will imply the equivalence of dealing with the approximation problem through its formulation as a minimization problem in (2.2) or through the discrete necessary conditions in (3.1). However, this still leaves the question of existence of $(u_n^*,x_n^*,\lambda_n^*)$ in N. For the remainder of this section, we will assume (A4) that (2.2) has a solution in N. Sufficient conditions are given later.

<u>Theorem 3.2</u>. Let (A1) - (A4) hold. Then for all λ_n in $S(\Delta_n,p)$ satisfying $\lambda_n(1) = 0$, all x_n in $S(\Delta_n,p)$ satisfying $x_n(0) = x_0$, and all u_n in $S(\Delta_n,p)$ so that (u_n,x_n,λ_n) is in N,

$$L_n(u_n^*,x_n^*,\lambda_n) = L_n(u_n^*,x_n^*,\lambda_n^*) \le L_n(u_n,x_n,\lambda_n^*).$$

<u>Proof</u>. The equality follows directly from (3.1)(a). Write $\Delta u_n = u_n - u_n^*$ and $\Delta x_n = x_n - x_n^*$ and expand $L_n(u_n,x_n,\lambda_n^*)$ to obtain

$$L_n(u_n,x_n,\lambda_n^*) = L_n(u_n^*,x_n^*,\lambda_n^*) + \sum_{i,j}{}' g_x(x_n^*,u_n^*,t)\Delta x_n$$

$$+ \sum_{i,j}{}' g_u(x_n^*,u_n^*,t)\Delta u_n + \sum_{i,j}{}' \int_0^1\{g_{xx}(x_n^* + s\Delta x_n,u_n^*,t)\Delta x_n^2$$

$$+ g_{uu}(x_n^*,u_n^* + s\Delta u_n,t)\Delta u_n^2\}(1-s)ds$$

$$(3.5) \qquad + \sum_{i,j}{}' \lambda_n^*(-\dot{\Delta}x_n + f_x(x_n^*,u_n^*,t)\Delta x_n + f_u(x_n^*,u_n^*,t)\Delta u_n)$$

$$+ \sum_{i,j}{}' \lambda_n^*(f_{xx}(x_n^* + s\Delta x_n,u_n^*,t)\Delta x_n^2 + f_{uu}(x_n^*,u_n^* + s\Delta u_n,t)\Delta u_n^2)(1-s)ds$$

$$+ \sum_{i,j}{}' (g_{xu}(x_n^* + s\Delta x_n,u_n^* + s\Delta u_n,t)\Delta x_n\Delta u_n + \lambda_n^* f_{xu}(x_n^* + s\Delta x_n,u_n^* + s\Delta u_n,t)$$

$$\Delta x_n\Delta u_n)(1-s)ds.$$

Now using A3 and arguing as in the proof of Theorem 3.1, (3.5) becomes

$$L_n(u_n,x_n,\lambda_n^*) \geq \sigma \sum_{i,j} (|\Delta x_n|^2 + |\Delta u_n|^2) + L_n(u_n^*,x_n^*,\lambda_n^*)$$

$$(3.6) \qquad + \sum_{i,j}{}' g_x(x_n^*,u_n^*,t)\Delta x_n + \sum_{i,j}{}' g_u(x_n^*,u_n^*,t)\Delta u_n$$

$$+ \sum_{i,j}{}' \dot{\lambda}_n^*\Delta x_n + \sum_{i,j}{}' \lambda_n^*(f_x(x_n^*,u_n^*,t)\Delta x_n + f_u(x_n^*,u_n^*,t)\Delta u_n).$$

Then using (3.1)(c) and (3.1)(e) we obtain

$$(3.7) \qquad L_n(u_n,x_n,\lambda_n^*) \geq \sigma \sum_{i,j}{}' (|\Delta x_n|^2 + |\Delta u_n|^2) + L_n(u_n^*,x_n^*,\lambda_n^*).$$

completing the proof.

Now if u_n and x_n are candidate solutions to the discrete problem (2.2), i.e. (2.2)(b) is satisfied, then $L_n(u_n,x_n,\lambda_n^*) = \sum_{i,j}{}' g(x_n,u_n,t)$. If $\hat{u}_n$ and $\hat{x}_n$ (with some λ_n) satisfy the necessary conditions (3.1), then using Theorem 3.2 we have $\sum_{i,j}{}' g(\hat{x}_n,\hat{u}_n,t) \leq \sum_{i,j}{}' g(x_n,u_n,t)$, i.e. $(\hat{x}_n,\hat{u}_n)$ solves (2.2). Thus (A3) implies the sufficiency of the necessary conditions. Moreover, from Eq(3.7) it follows that the values of x_n and u_n are unique at the collocation points ξ_{ij}. Since $x_n(0) = x_0, x_n$ is unique.

Now that we have shown the equivalence of dealing with the discrete problem either directly or through the discrete necessary conditions, convergence theorems can be established by an analysis of (3.1) as a discretization of (2.1).

Lemma 3.3 The equation $g_u(x,u,t) + f_u(x,u,t)\lambda = 0$ can be solved uniquely for $u = \phi(x,\lambda,t)$ in a neighborhood N' of (u^*,x^*,λ^*) so that $u^* = \phi(x^*,\lambda^*,t)$. Moreover, if P is in C^ν, ϕ has ν continuous derivatives.

Proof. This is a consequence of (A3) and the implicit function theorem.

Using Lemma 3.3, the necessary conditions may be written as the two-point boundary value problem

$$\text{(a)} \quad \dot{\lambda}^* + f_x(x^*,\phi(x^*,\lambda^*,t),t)\lambda^* + g_x(x^*,\phi(x^*,\lambda^*,t),t) = 0$$

$$(3.8)\text{(b)} \quad -\dot{x}^* + f(x^*,\phi(x^*,\lambda^*,t),t) = 0$$

$$\text{(c)} \quad x^*(0) = x_0, \quad \lambda^*(1) = 0.$$

Also using Lemma 3.3, the necessary conditions derived in Theorem 3.1 can be written as

$$\text{(a)}\quad \dot{x}_n^*(\xi_{ij}) = f(x_n^*(\xi_{ij}),\phi(x_n^*(\xi_{ij}),\lambda_n^*(\xi_{ij}),\xi_{ij}),\xi_{ij})$$

$$\text{(b)}\quad \dot{\lambda}_n^*(\xi_{ij}) = -f_x(x_n^*(\xi_{ij}),\phi(x_n^*(\xi_{ij})\lambda_n^*(\xi_{ij}),\xi_{ij}),\xi_{ij})\lambda_n^*(\xi_{ij})$$

(3.9) $\qquad i=1,\ldots,n;\ j=1,\ldots,p$

$$-g_x(x_n^*(\xi_{ij}),\phi(x_n^*(\xi_{ij}),\lambda_n^*(\xi_{ij}),\xi_{ij}),\xi_{ij})$$

$$\text{(c)}\quad x_n^*(0) = x_0,\quad \lambda_n^*(1) = 0.$$

It follows that Eq(3.9) represents the equations for collocation at Gauss points as an approximation scheme for the solution to the necessary conditions (3.8). With one additional assumption, we can now appeal to the theory of collocation methods for vector systems as given in, for example, Russell [8] and deduce convergence and convergence rates. We add the assumption (A5) that when Eq (3.8) is linearized about (x^*,λ^*), the resulting problem is uniquely solvable. See [8] for details.

Theorem 3.4. Let hypotheses (A1)-(A3) and (A5) hold. Then solutions x_n^*,λ_n^* to Eq 3.9 exist and are unique in a neighborhood of (x^*,λ^*) for all partitions Δ_n with $|\Delta_n|$ sufficiently small.

If (P) is in C^{2p}, then

$$||x_n^*-x^*||_\infty = O(|\Delta_n|^{(p+1)})$$

$$||\lambda_n^*-\lambda^*||_\infty = O(|\Delta_n|^{(p+1)})$$

$$|x_n^*(t_i) - x^*(t_i)| = O(|\Delta_n|^{2p})$$

and

$$|\lambda_n^*(t_i) - \lambda^*(t_i)| = O(|\Delta_n|^{2p})$$

where $t_i \in \Delta_n$.

If we define $u_n = \phi(x_n^*,\lambda_n^*,t)$, it follows from Theorem 3.4 and Lemma 3.3 that u_n-u^* will satisfy the bounds of Theorem 3.4. Of course u_n is not necessarily identically u_n^*, but $u_n(\xi_{ij}) = u_n^*(\xi_{ij})$, i.e., they agree at the collocation points.

The error in the approximation found by collocating at Gauss points has been discovered to be expressible by an expansion in which the main term is local [3], [9]. For the system (3.8), these results imply (assuming sufficient smoothness) that

$$(x^* - x_n^*)(t) = \frac{(D^{p+1}x^*)(t_i)}{2^{p+1}}\, P\left(\frac{2}{h_i}(t-t_{i+\frac{1}{2}})\right)h_i^{p+1} + O(|\Delta_n|^{p+2}) \qquad (3.10)$$

for t in $[t_i,t_{i+1}]$ where $t_{i+\frac{1}{2}} = (t_i + t_{i+1})/2$, $h_i = t_{i+1} - t_i$ and

$$P(\xi) = \frac{d^{p-1}}{d\xi^{p-1}}\,\frac{(\xi^2-1)^p}{(2p)!}$$

ıor ξ in $(-1,1)$. A similar expression is valid for $\lambda^*-\lambda^*_n$. Using (3.10), an estimate for $(D^{p+1}x^*)(t_i)$ provides an estimate for the local error. Several procedures have been suggested for producing this estimate. One can [9] approximate $(D^{p+1}x^*)(t)$ by differentiating the piecewise linear function that interpolates $D^p x^*_n$ at $t_{i+\frac{1}{2}}$ for each i. Then based on the expansion (3.10), a natural way of selecting an improved partition would be to (asymptotically) minimize the error by equidistributing the local term, i.e., by requiring $(h^*_i)^{p+1}|(D^{p+1}x^*)(t_i)|2^{-p-1}||P|| = C$, all i. Details on the implementation of this and other approaches to mesh selection can be found in [3] and [9].

An analogous expression for the error in the control is more complicated because it is only implicitly defined by (3.8). However, we may write $(u_n-u^*)(t) = \phi(x^*_n,\lambda^*_n,t) - \phi(x^*,\lambda^*,t) = \phi_x(x^*,\lambda^*,t) \cdot (x^*_n - x^*) + \phi_\lambda(x^*,\lambda^*,t) \cdot (\lambda^*_n - \lambda^*) +$ (higher order terms). Writing $\phi^*_x(t) = \phi_x(x^*,\lambda^*,t)$ and $\phi^*_\lambda(t) = \phi_\lambda(x^*,\lambda^*,t)$, we then have

$$(u_n-u^*)(t_i) = \phi^*_x(t_i)\,\frac{(D^{p+1}x^*)(t_i)}{2^{p+1}}\,P(\frac{2}{h_i}(t-t_{i+\frac{1}{2}}))h_i^{p+1}$$

$$+\ \phi^*_\lambda(t_i)\,\frac{(D^{p+1}\lambda^*)(t_i)}{2^{p+1}}\,P(\frac{2}{h_i}(t-t_{i+\frac{1}{2}}))h_i^{p+1} + O(|\Delta_n|^{p+2})$$

The term $(D^{p+1}x^*)(t_i)$ can be estimated as above. Computing based on (2.2), neither ϕ nor λ^*_n will be explicitly available. However one could solve (3.1)(e) for values of λ^*_n and then differentiate a local interpolant to produce an approximation to $(D^{p+1}\lambda^*)(t_i)$. In certain cases (2.1(a) can provide an explicit formula that can provide values. Writing $u = \phi(x,\lambda,t)$, Eq. (2.1)(a) can be differentiated with respect to x and λ and the resulting equations evaluated using the computed u^*_n, x^*_n and λ^*_n to obtain approximate values for ϕ^*_x and ϕ^*_λ. For example, $\phi_x(x,u,t) = -(g_u x(x,u,t) + f_{ux}(x,u,t)\lambda)/(g_{uu}(x,u,t) + f_{uu}(x,u,t)\lambda)$. We are currently performing numerical experiments using these ideas. The results will be reported elsewhere.

Finally, we present the results of a simple numerical experiment indicating the possibility of obtaining good accuracy. We treated the problem

$$\min \int_0^1 (u^2+x^2)dt$$

$$\text{S.T.}\quad \dot{x} = x+u,\quad x(0) = 1.$$

We chose $p = 2$ so that continuous quadratic splines were the approximating functions. Thus there were two Gauss points in each subinterval. For $|\Delta_n| = 1/2, 1/4$ and $1/8$ (uniform meshes) the errors at $t = 1$ for x^* were respectively $.304 \cdot 10^{-3}$, $.211 \cdot 10^{-4}$ and $.117 \cdot 10^{-5}$. The solution $x^*(1) = 1.2347436851$. We used Armijo's implementation of steepest descent and the computation terminated when the sum of partials was less than 10^{-8} (double precision).

References

1. W.E. Bosarge, Jr., O.G. Johnson, R.S. Mcknight and W.P. Timlake, The Ritz-Galerkin procedure for nonlinear control problems, SIAM J. Numer. Anal. 10 (1973), 94-111.

2. W.E. Bosarge, Jr. and O.G. Johnson, Error bounds of high order accuracy for the state regulator problem via piecewise polynomial approximations, SIAM J. Control 9 (1971), 15-28.

3. C. deBoor, Good approximation by splines with variable knots. II, Conference on the Numerical Solution of Differential Equations, Lecture Notes in Mathematics, Vol. 363, Springer-Verlag, New York.

4. Richard S. Falk, Approximation of a class of optimal control problems with order of convergence estimates, J. Math. Anal. Appl. 44 (1973), 28-47.

5. William W. Hager, The Ritz-Trefftz method for state and control constrained optimal control problems, SIAM J. Numer. Anal. 12 (1975), 854-867.

6. F.H. Mathis and G.W. Reddien, Ritz-Trefftz approximations in optimal control, SIAM J. Control and Optimization, to appear.

7. G.W. Reddien, Collocation at Gauss points as a discretization in optimal control, SIAM J. Control and Optimization, to appear.

8. R.D. Russell, Collocation for systems of boundary value problems, Numer. Math. 23 (1974), 119-133.

9. R.D. Russell and J. Christiansen, Adaptive mesh selection strategies for solving boundary value problems, SIAM J. Numer. Anal. 15 (1978), 59-80.

ON THE STABILITY OF CERTAIN DISCRETIZATIONS ON NONUNIFORM MESHES

Eusebius J.Doedel
Mathematics Department
Vanderbilt University
Nashville, Tennessee 37235

ABSTRACT. A class of finite difference collocation methods is shown to be stable on general nonuniform meshes without the assumption of a bounded mesh ratio. Some error estimates are included.

1. INTRODUCTION. We consider the convergence of certain discrete approximations to boundary value problems in ordinary differential equations. The primary object of study is the stability of these approximations, especially on nonuniform meshes. To investigate this stability we employ a technique used previously by Kreiss (1972) for analyzing the stability of general difference methods on uniform meshes.

A simple example is treated in Section 2. The general stability results for a large class of compact approximation methods are presented in Section 3. The approximations are of a type studied previously by, for example, Osborne (1964),(1974), Doedel (1978),(1979), Lynch and Rice (1976). Exactly the same analysis could be used for other methods such as for example collocation methods with C^{n-1} piecewise polynomials for solving nth-order equations. Convergence of the latter type of discretization has been extensively studied. (See for example the survey paper by Reddien.)

Throughout only linear problems will be considered. That this poses no real restriction is justified in, e.g., Pereyra (1967) and Keller (1975).

2. AN EXAMPLE. Consider the simple second-order equation

$$\text{(2.1)} \qquad Ly \equiv y'' + b(x)y = f(x), \quad 0 \le x \le 1,$$

subject to

$$\text{(2.1a)} \qquad y(0) = y(1) = 0,$$

where, for notational convenience only, we have assumed the y' term to be absent. Introduce a mesh $\{0 = x_0 < x_1 < \ldots < x_J = 1\}$, with

$$h_j \equiv x_j - x_{j-1}, \; h \equiv (h_1, h_2, \ldots, h_J) \text{ and } \quad |h| \equiv \max_j h_j .$$

A finite difference approximation to (2.1) that is second-order accurate, even if the mesh is not uniform, and that requires evaluation of b and f at one point only, (see Doedel (1978), Example 5.1), is given by

$$(2.2)\qquad 2(h_j+h_{j+1})^{-1}[h_{j+1}^{-1}(u_{j+1}-u_j) - h_j^{-1}(u_j-u_{j-1})] +$$

$$b(z_j)[9h_jh_{j+1}(h_j+h_{j+1})]^{-1}(c_{-1}u_{j-1}+c_0u_j+c_1u_{j+1}) =$$

$$= f(z_j),$$

where

$$z_j = x_j + \tfrac{1}{3}(h_{j+1}-h_j),\quad c_{-1} = h_{j+1}(h_j-h_{j+1})(2h_{j+1}+h_j),$$

$$c_0 = (h_j+h_{j+1})(2h_j+h_{j+1})(h_j+2h_{j+1}) \text{ and } c_1= h_j(h_{j+1}-h_j)(2h_j+h_{j+1}).$$

The above approximation leads to a linear system of equations of the form

$$L_hu_h = f_h,\ \text{where } u_h \equiv (u_1,u_2,\ \ldots,\ u_{J-1})^T.$$

We have

$$||L_hy_h-f_h||_\infty = O(|h|^2),$$

and convergence of u_h to y_h with second-order accuracy follows, provided L_h is stable, i.e., provided that L_h has a uniformly bounded inverse. If $b(x) \leq b_* < 0$ then this is easily verified using, for example, the Banach Lemma. However, in general, such an algebraic approach can be rather complicated, if at all possible.

It can be shown (Doedel (1979)), that the difference approximation (2.2) is equivalent to the following collocation method: Corresponding to each interior meshpoint x_j find a polynomial $p_j \in P_2$ of degree at most 2 satisfying the collocation equations

$$(2.3)\qquad Lp_j(z_j) = f(z_j),\qquad 1 \leq j \leq J-1,$$

the matching conditions

$$(2.4a)\qquad p_j(x_{j+i}) = p_{j+1}(x_{j+i}),\ i = 0,1,\qquad 1 \leq j \leq J-2,$$

and the boundary conditions

(2.4b) $$p_1(0) = p_{J-1}(1) = 0.$$

The above observation suggests alternate ways of analyzing (2.2), such as, for example, via projection techniques. This is not exactly the approach we shall take here however. Instead a closely related technique used by Kreiss (1972) will be employed.

Let P_h^* denote the space of all

$$\underline{p}_h(x) \equiv \{p_j(x)\}_{j=1}^{J-1}, \quad p_j \in P_2,$$

with the $p_j(x)$ satisfying (2.4a,b). For $\underline{p}_h \in P_h^*$ introduce the norm

$$||\underline{p}_h||_2 = \max_{0\leq \ell\leq 2} \max_{1\leq j\leq J-1} \max_{[x_{j-1},x_{j+1}]} |p_j^{(\ell)}(x)|.$$

Also for $p \in C^2[0,1]$ and $\underline{p}_h \in P_h^*$ it will be convenient to use the notation

$$||\underline{p}_h - p||_k \equiv \max_{0\leq \ell\leq k} \max_{1\leq j\leq J-1} \max_{[x_{j-1},x_{j+1}]} |p_j^{(\ell)}(x) - p^{(\ell)}(x)|.$$

Indeed, the above could serve as norm in an appropriate Banach space structure. However, such a formal set-up is not strictly necessary in subsequent analysis.

First we need the following:

Lemma 2.1 Let $\{h^\nu\}_{\nu=1}^{\infty}$ be a sequence of meshes with $|h^\nu| \to 0$ as $\nu \to \infty$. For each ν let $\underline{p}_{h^\nu} \in P_{h^\nu}^*$, with $||\underline{p}_{h^\nu}||_2 = 1$.

Then there is a subsequence $\{\underline{p}_{h^\nu}\}_{\nu=1}^{\infty}$

and a function $p \in C^1[0,1]$ such that

$$||\underline{p}_{h^\nu} - p||_1 \to 0 \quad \text{as} \quad \nu \to \infty.$$

Remark For notational simplicity we do not distinguish between sequence and subsequence. This can be justified by relabelling the original sequence.

Proof   By simply restricting the domains of each of the component polynomials

$$\{p_j\}_{j=1}^{J-2} \text{ of } \underline{p}_{h_\nu} \text{ to } [x_{j-1}, x_j]$$

respectively, we can extract a single valued continuous function $\tilde{\underline{p}}_{h_\nu}$ from each $\underline{p}_{h_\nu}$. Also $||\underline{p}_{h_\nu}||_2 = 1$ implies in particular that

$$\max_j \max_{[x_{j-1}, x_{j+1}]} |p_j'(x)| \leq 1.$$

Thus the sequence $\{\tilde{\underline{p}}_{h_\nu}\}_{\nu=1}^{\infty}$ is equicontinuous and therefore has a subsequence

$$\tilde{\underline{p}}_{h_\nu} \to p \in C[0,1].$$

In fact, using the boundedness of the p_j', it is easy to see that the multivalued function $\underline{p}_{h_\nu}$ converges to p, i.e.,

$$||\underline{p}_{h_\nu} - p||_0 \to 0 \quad \text{as} \quad \nu \to \infty.$$

We want to show that $p \in C^1[0,1]$. Define

$$\underline{q}_{h_\nu} \equiv \underline{p}'_{h_\nu} \equiv \{p_j'\}_{j=1}^{J-1}.$$

From Rolle's Theorem and the matching conditions (2.4a) it follows that every two consecutive component polynomials p_j and p_{j+1} of a given $\underline{p}_{h_\nu}$ satisfy

$$p_j'(\xi_j) = p_{j+1}'(\xi_j) \text{ for some } \xi_j \in (x_j, x_{j+1}).$$

(See Fig.2.1). Therefore from each $\underline{q}_{h_\nu}$ we can extract a single valued continuous function $\tilde{\underline{q}}_{h_\nu}$.

Tne sequence $\{\tilde{\underline{q}}_{h_\nu}\}_{\nu=1}^{\infty}$ is equicontinuous since

$$\max_j \max_{[x_{j-1}, x_{j+1}]} |p_j'(x)| \leq 1.$$

Hence there is a subsequence $\tilde{\underline{q}}_{h^\nu} \to q \in C[0,1]$. Again convergence of the multivalued $\underline{q}_{h^\nu}$ to q follows from the boundedness of the q_j'. Thus

$$||\underline{q}_{h^\nu} - q||_0 \to 0 \quad \text{as} \quad \nu \to \infty.$$

We contend that $q = p'$. Let $x \in [0,1]$. Then

$$\int_0^x \tilde{\underline{q}}_{h^\nu} = \int_0^{\xi_1} \tilde{\underline{q}}_{h^\nu} + \sum_{j=1}^{j[x]-1} \int_{\xi_j}^{\xi_{j+1}} \tilde{\underline{q}}_{h^\nu} + \int_{\xi_{j[x]}}^{x} \tilde{\underline{q}}_{h^\nu}.$$

(Here, for each ν, $j[x]$ is the largest j for which $\xi_{j[x]} < x$.)

By definition of $\underline{q}_{h^\nu}$ this implies

$$\int_0^x \tilde{\underline{q}}_{h^\nu} = \int_0^{\xi_1} p_1' + \sum_{j=1}^{j[x]-1} \int_{\xi_j}^{\xi_{j+1}} p_{j+1}' + \int_{\xi[x]}^{x} p'_{j[x]+1} =$$

$$= p_{j[x]+1}(x) - \sum_{j=1}^{j[x]} [p_{j+1}(\xi_j) - p_j(\xi_j)] - p_1(0) =$$

$$= \tilde{\underline{p}}_{h^\nu}(x) - \sum_{j=1}^{j[x]} (\xi_j - x_j)[q_{j+1}(\eta_{j,1}) - q_j(\eta_{j,2})] -$$

$$- \tilde{\underline{p}}_{h^\nu}(0),$$

where $\eta_{j,1}, \eta_{j,2} \in (x_j, \xi_j)$. It follows that

$$\left| \int_0^x \tilde{\underline{q}}_{h^\nu} - [\tilde{\underline{p}}_{h^\nu}(x) - \tilde{\underline{p}}_{h^\nu}(0)] \right| \le \max_j |q_{j+1}(\eta_{j,1}) - q_j(\eta_{j,2})| =$$

$$= \max_j |\eta_{j,1} - \eta_{j,2}|\,|q_j'(x_j)| \le |h^\nu|,$$

where we used the fact that $q_j' = p_j''$ is constant, since $p_j \in P_2$. Letting $\nu \to \infty$ and differentiating gives $p'(x) = q(x)$. It also follows that

$$||\underline{p}_{h_\nu} - p||_1 \to 0 \quad \text{as} \quad \nu \to \infty \; . \; \square$$

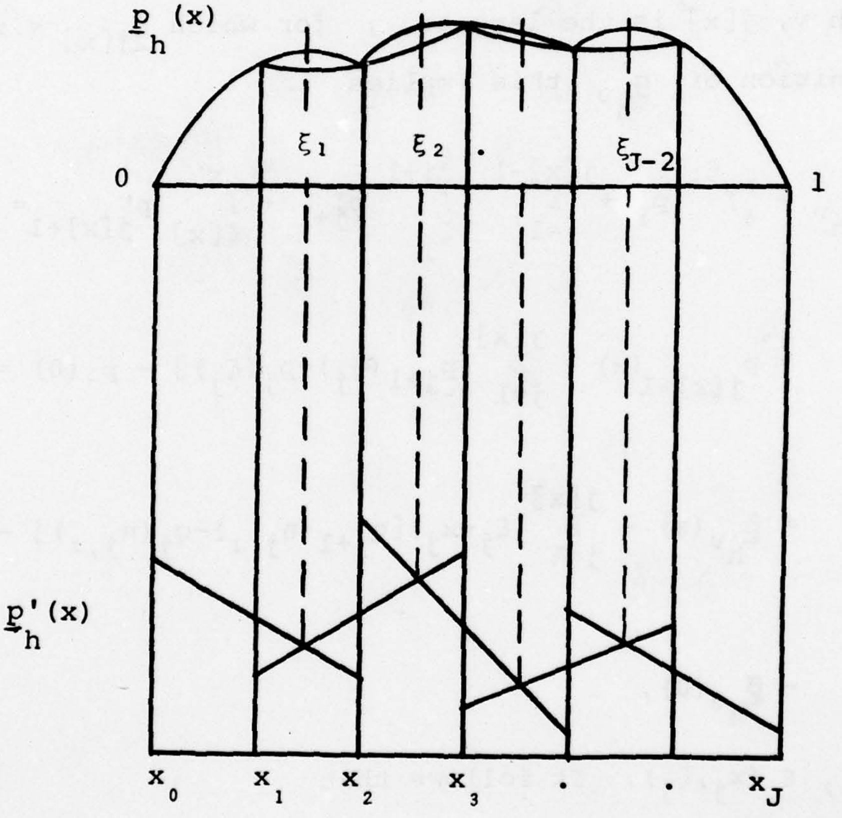


FIGURE 2.1

We can now prove:

Theorem 2.2

Let the homogeneous problem corresponding to (2.1),(2.1a) only admit the zero solution. Let $b(x) \in C[0,1]$. Then there exist positive constants K and δ, such that the equations (2.3),(2.4a,b) admit a unique solution $\underline{p}_h \in P_h^*$ and such that

$$(2.5) \qquad ||\underline{p}_h||_2 \leq K \max_j |f(z_j)|,$$

whenever $|h| \in (0,\delta]$.

Proof

If (2.3),(2.4a,b) do not have a unique solution $\underline{p}_h \in P_h^*$ for all small h, then, since $\dim P_h^* = J-1$ equals the number of equations in (2.3), we can find a sequence of meshes

$$\{h^\nu\}_{\nu=1}^\infty, \text{ with } |h^\nu| \to 0 \quad \text{as} \quad \nu \to \infty,$$

and corresponding

$$\underline{p}_{h^\nu} \in P^*_{h^\nu}, \quad \text{with } ||\underline{p}_{h^\nu}||_2 = 1,$$

such that $\underline{p}_{h^\nu}$ satisfies the homogeneous equations corresponding to (2.3).

By Lemma 2.1, there is a subsequence $\{\underline{p}_{h^\nu}\}_{\nu=1}^\infty$

and a function $p \in C^1[0,1]$ such that

$$||\underline{p}_{h^\nu}-p||_1 \to 0 \quad \text{as} \quad \nu \to \infty.$$

Using the collocation equations (2.3) we can show that in fact $p \in C^2[0,1]$ and that $Lp = 0$, but $p \not\equiv 0$. This is a contradiction of the first assumption of the theorem. Hence existence has been established. The actual details of this argument are essentially the same as those required for establishing the bound (2.5). Therefore we will only give details for the latter.

Assuming the existence of $\underline{p}_h$ for all sufficiently small $|h|$, suppose that (2.5) fails to hold. Then again, we can find a sequence of meshes

$$\{h^\nu\}_{\nu=1}^\infty, \quad \text{with} \quad |h^\nu| \to 0 \quad \text{as} \quad \nu \to \infty,$$

and for each mesh quantities

$$\{f^\nu(z_j)\} \quad \text{with} \quad \max_j |f^\nu(z_j)| \to 0,$$

such that the corresponding unique solutions

$$\underline{p}_{h^\nu} \in P_h^*$$

of (2.3), (2.4a,b) satisfy

$$||\underline{p}_{h^\nu}||_2 = 1.$$

Thus (as in the existence part) by Lemma 2.1, there is a subsequence

$$\{\underline{p}_{h^\nu}\}_{\nu=1}^\infty$$

and a function $p \in C^1[0,1]$ such that

$$||\underline{p}_{h^\nu} - p||_1 \to 0 \quad \text{as} \quad \nu \to \infty.$$

Again we claim that p satisfies the homogeneous equations (2.2a,b).

Let $s \in [0,1]$ and consider the subsequence

$$\{\underline{p}_{h^\nu}\}_{\nu=1}^\infty.$$

For each ν let j_ν be such that $s \in [x_{j_\nu-1}, x_{j_\nu+1}]$. Then, since $p_{j_\nu} \in P_2$, we have:

$$(2.6) \qquad |p''_{j_\nu}(s) + b(s)p(s)| = |p''_{j_\nu}(z_{j_\nu}) + b(s)p(s)|$$

$$= |f^\nu(z_{j_\nu}) - b(z_{j_\nu})p_{j_\nu}(z_{j_\nu}) + b(s)p(s)|$$

$$\leq |f^\nu(z_{j_\nu})| + |b(s)p(s) - b(s)p_{j_\nu}(s)| +$$

$$|b(s)p_{j_\nu}(s) - b(s)p_{j_\nu}(z_{j_\nu})|$$

$$+ |b(s)p_{j_\nu}(z_{j_\nu}) - b(z_{j_\nu})p_{j_\nu}(z_{j_\nu})|$$

$$\leq |f^\nu(z_{j_\nu})| + |b(s)||p(s) - p_{j_\nu}(s)|$$

$$+ |b(s)||p_{j_\nu}(s) - p_{j_\nu}(z_{j_\nu})| + |p_{j_\nu}(z_{j_\nu})||b(s) - b(z_{j_\nu})|$$

$$\leq |f^\nu(z_{j_\nu})| + ||b||_\infty\{||\underline{p}_{h^\nu} - p||_1 + 2|h^\nu|\}$$

$$+ |b(s) - b(z_{j_\nu})| \to 0$$

as $\nu \to \infty$.

Since b is continuous this convergence is in fact uniform in s.

Similar to the procedure in the proof of Lemma 2.1, one can extract a single valued continuous function

$$\underline{\tilde{p}}'_{h^\nu} \text{ from each } \underline{p}'_{h^\nu}$$

and we have

$$\int_0^x \underline{\tilde{p}}''_{h^\nu}(s)ds + \int_0^x b(s)p(s) \to 0 \text{ as } \nu \to \infty.$$

(Note however that $\underline{\tilde{p}}''_{h^\nu}$ need not be continuous.) Upon integration, using the continuity of $\underline{\tilde{p}}'_{h^\nu}$, it follows that

$$\tilde{p}'_{h_\nu}(x) - \tilde{p}'_{h_\nu}(0) + \int_0^x b(s)p(s)dx \to 0 \text{ as } \nu \to \infty.$$

Taking the limit yields

$$p'(x) - p'(0) + \int_0^x b(s)p(s)ds = 0.$$

This implies in particular that $p \in C^1[0,1]$. Differentiation gives

$$p''(x) + b(x)p(x) = 0 \tag{2.7}$$

It is clear that p also satisfies the homogeneous boundary conditions (2.1a). Further with the aid of (2.6) and (2.7) we have

$$||p_{h_\nu} - p||_2 \to 0 \quad \text{as} \quad \nu \to \infty.$$

Since

$$||p_{h_\nu}||_2 = 1$$

this implies that $p \not\equiv 0$. Hence a contradiction has been obtained.□

Remark

For any reasonable approximation space P_h^*, (i.e., a space whose elements satisfy appropriate matching or continuity conditions), it is easy to construct a sequence

$$p_{h_\nu} \to p \in C^1[0,1],$$

as in Lemma 2.1. The difficulty lies in showing that $p \in C^2[0,1]$. In Theorem 2.2 this was accomplished quite easily, due to the fact that

$$p_j''(s) \equiv p_j''(z_j) \text{ since } p_j \in P_2.$$

In other words, the second derivative of any given polynomial component p_j is completely determined by the value of the second derivative at the collocation point corresponding to p_j. This local dependence is what characterizes so-called "compact" difference approximations. It is also characteristic of collocation procedures with C^1 piecewise polynomials for solving second-order equations.

More generally, for $n^{\underline{th}}$ order differential equations, a complete stability analysis is relatively simple if the "degree of matching" of the piecewise polynomial spaces is equal to n-1. The result of this analysis is essentially that such discretizations are unconditionally stable.

3. GENERAL STABILITY RESULTS

Consider the $n^{\underline{th}}$ order equation

$$\text{(3.1a)} \quad Ly \equiv y^{(n)}(x) + \sum_{\ell=o}^{n-1} a_\ell(x)y^{(\ell)}(x) = f(x),\ 0 \le x \le 1,$$

subject to boundary conditions of the form

$$\text{(3.1b)} \quad B_k y \equiv \sum_{\ell=o}^{n-1} \alpha_{k,\ell}(0)y^{(k)}(0) + \sum_{\ell=o}^{n-1} \alpha_{k,\ell}(1)y^{(k)}(1) =$$

$$= \beta_k,\ 1 \le k \le n.$$

Reference will also be made to the corresponding homogeneous problem

$$\text{(3.2a)} \qquad Ly = 0,\quad 0 \le x \le 1,$$

subject to

$$\text{(3.2b)} \qquad B_k y = 0,\quad 1 \le k \le n.$$

We approximate (3.1) by a class of generalized finite difference methods. These methods can be interpreted as collocation methods in the following manner: Define a mesh as before. To each meshpoint x_j, $(0 \le j \le J-n)$, associate a polynomial

$$p_j \in P_{n+m-1}. \text{ (More accurately: } p_{h,j}(x).)$$

We think of the interval $[x_j, x_{j+n}]$ as the domain of p_j. Let

$$\underline{p}_h \equiv \{p_j\}_{j=0}^{J-n}.$$

Thus $\underline{p}_h(x)$ is defined on $[0,1]$. Note however, that $\underline{p}_h$ is not single valued. The discrete method now consists of finding $\underline{p}_h$ satisfying the collocation equations

$$(3.3) \qquad Lp_j(z_{j,i}) = f(z_{j,i}), \; 1 \le i \le m, \; 0 \le j \le J-n,$$

where for each j the

$$z_{j,i} \text{ are distinct points in } [x_j, x_{j+n}].$$

In addition $\underline{p}_h$ is required to satisfy the matching conditions

$$(3.4) \qquad p_j(x_{j+i}) = p_{j+1}(x_{j+i}), \; 1 \le i \le n, \; 0 \le j \le J-n-1,$$

as well as the boundary conditions

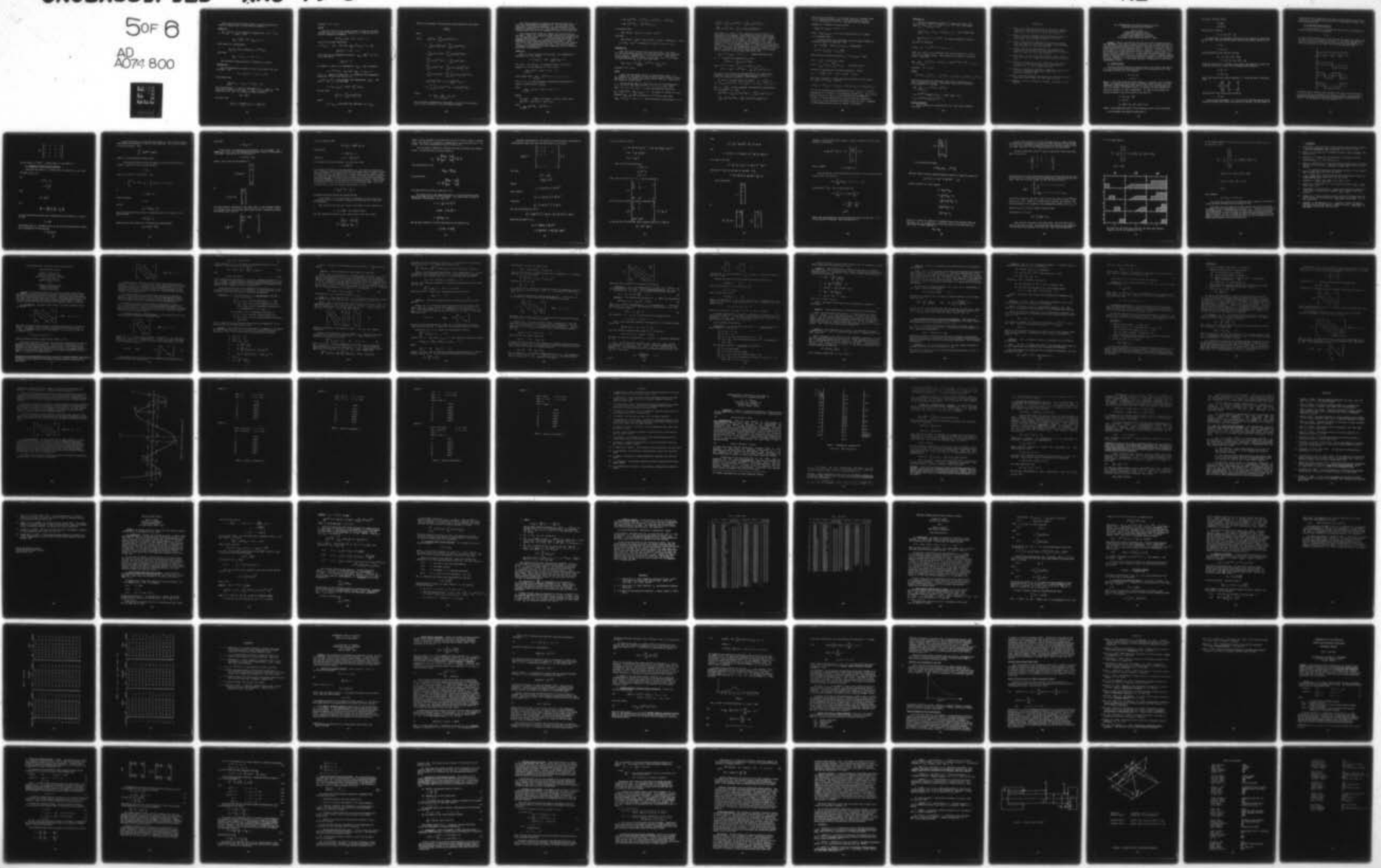
$$(3.5) \qquad B_k\underline{p}_h = \beta_k \qquad 1 \le k \le n.$$

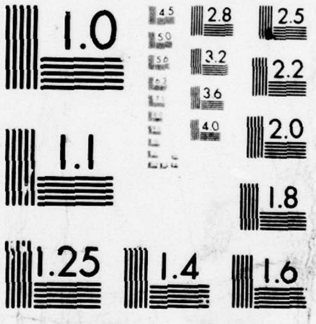
For each fixed mesh let $P_h^{n,m}$ denote the space of all $\underline{p}_h$ satisfying (3.4), and define

$$||\underline{p}_h - p||_k \equiv \max_{0\le \ell \le k} \; \max_{0 \le j \le J-n} \; \max_{[x_j, x_{j+n}]} |p_j^{(\ell)}(x) - p^{(\ell)}(x)|.$$

AD-A074 800 ARMY RESEARCH OFFICE RESEARCH TRIANGLE PARK NC F/G 5/9
PROCEEDINGS OF THE 1979 ARMY NUMERICAL ANALYSIS AND COMPUTERS C--ETC(U)
SEP 79
UNCLASSIFIED ARO-79-3 NL

5 OF 6
AD
A074 800





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

The proof of the following lemma is easily obtained by inductive application of the proof of Lemma 2.1:

Lemma 3.1

Let $\{h^\nu\}_{\nu=1}^{\infty}$ be a sequence of meshes with $|h^\nu| \to 0$ as $\nu \to \infty$. For each ν let

$$\underline{p}_{h^\nu} \in P_{h^\nu}^{n,m}, \text{ with } ||\underline{p}_{h^\nu}||_n = 1.$$

Then there is a subsequence

$$\{\underline{p}_{h^\nu}\}_{\nu=1}^{\infty} \text{ and a function } p \in C^{n-1}[0,1],$$

such that

$$||\underline{p}_{h^\nu} - p||_{n-1} \to 0 \quad \text{as} \quad \nu \to \infty.$$

The following generalization of Theorem 2.2 holds:

Theorem 3.2

Let the homogeneous problem (3.2a,b) only admit the zero solution. Let

$$a_\ell \in C[0,1], \quad \ell = 0,1,\ldots,n-1,$$

and assume that

$$\min_{i_1 \neq i_2} |z_{j,i_1} - z_{j,i_2}| \geq c|x_{j+n} - x_j|,$$

for some constant c that is independent of j and h. Then there exist positive constants K and δ, such that the equations (3.3) and (3.5) admit a unique solution

$$\underline{p}_h \in P_h^{n,m}$$

and such that

$$||\underline{p}_h||_n \leq K\{\max_{i,j}|f(z_{j,i})| + \max_{\ell}|\beta_\ell|\},$$

whenever $|h| \in (0,\delta]$.

Proof:

The first part of the proof closely follows the corresponding part of the proof of Theorem 2.2. Thus we can construct a sequence of meshes

$$\{h^\nu\}_{\nu=1}^{\infty}, \text{ with } |h^\nu| \to 0 \text{ as } \nu \to \infty,$$

and for each mesh quantities

$$\{f^\nu(z_{j,i})\} \quad \text{and} \quad \{\beta_\ell^\nu\} \text{ with } \max_{i,j} |f^\nu(z_{j,i})| \to 0 \quad \text{and}$$

$$\max_\ell |\beta_\ell^\nu| \to 0 \quad \text{as} \quad \nu \to \infty,$$

such that the corresponding solution $\underline{p}_{h^\nu} \in P_h^{n,m}$ of (3.3), (3.5) satisfies

$$||\underline{p}_{h^\nu}||_n = 1.$$

By Lemma 3.1 there is a subsequence $\{\underline{p}_{h^\nu}\}$ and a function

$$p \in C^{n-1}[0,1] \quad \text{such that } ||\underline{p}_{h^\nu} - p||_{n-1} \to 0$$

as $\nu \to \infty$. Again we claim that p satisfies the homogeneous equations (3.2a,b) and that $p \not\equiv 0$.

Let $s \in [0,1]$ and consider the subsequence $\{\underline{p}_{h^\nu}\}$. For each ν let j_ν be such that

$$s \in [x_{j_\nu}, x_{j_\nu + n}]. \text{ Since } p_{j_\nu}^{(n)} \in P_{m-1}$$

we can write

$$p_{j_\nu}^{(n)}(x) = \sum_{i=1}^{m} \psi_i(x) p_{j_\nu}^{(n)}(z_i^\nu),$$

where

$$z_i^\nu \equiv z_{j_\nu,i}, \text{ and where the functions } \psi_i \equiv \psi_{j_\nu,i}$$

denote the Lagrange interpolating coefficients for the points

$$\{z_i^\nu\}_{i=1}^m .$$

Then

$$(3.6) \qquad |p_{j_\nu}^{(n)}(s) + \sum_{\ell=o}^{n-1} a_\ell(s) p^{(\ell)}(s)| =$$

$$= |\sum_{i=1}^{m} \psi_i(s) p_{j_\nu}^{(n)}(z_i^\nu) + \sum_{\ell=o}^{n-1} a_\ell(s) p^{(\ell)}(s)|$$

$$= |\sum_{i=1}^{m} \psi_i(s)\{f^\nu(z_i^\nu) - \sum_{\ell=o}^{n-1} a_\ell(z_i^\nu) p_{j_\nu}^{(\ell)}(z_i^\nu)\} +$$

$$+ \sum_{\ell=o}^{n-1} a_\ell(s) p^{(\ell)}(s)| \leq |\sum_{i=1}^{m} \psi_i(s) f^\nu(z_i^\nu)| +$$

$$+ \sum_{\ell=o}^{n-1} |a_\ell(s) p^{(\ell)}(s) - \sum_{i=1}^{m} \psi_i(s) a_\ell(z_i^\nu) p_{j_\nu}^{(\ell)}(z_i^\nu)|$$

$$= |\sum_{i=1}^{m} \psi_i(s) f^\nu(z_i^\nu)| + \sum_{\ell=o}^{n-1} |\sum_{i=1}^{m} \psi_i(s)[a_\ell(s) p^{(\ell)}(s) -$$

$$- a_\ell(z_i^\nu) p_{j_\nu}^{(\ell)}(z_i^\nu)]| \leq m\, K_1 \max_{i,j} |f^\nu(z_{j,i})| +$$

$$+ K_1 \sum_{\ell=o}^{n-1} \sum_{i=1}^{m} \{|a_\ell(s) p^{(\ell)}(s) - a_\ell(z_i^\nu) p_{j_\nu}^{(\ell)}(z_i^\nu)|\},$$

where

$$K_1 \equiv \max_{i,j} \max_{[x_j, x_{j+n}]} |\psi_{j,i}(x)|$$

can be chosen independent of the mesh, in view of the restriction on the location of the collocation points.

This final expression becomes arbitrarily small as $\nu \to \infty$ and this convergence is uniform in s. In fact the terms appearing in the double summation can be estimated in the manner of inequality (2.6). The remainder of the proof now proceeds much like that of Theorem 2.2 and will be omitted.□

The theorem above does not impose any restrictions on the mesh. There is a condition that involves the location of the collocation points $z_{j,i}$. These points are assumed to be locally semi-uniform. Even this restriction can be removed by requiring some additional continuity. More continuity also allows bounding derivatives of order greater than n in the approximate solution. For proving these statements we need the following elementary fact concerning polynomial interpolation.

Lemma 3.3

Let $s \in [0,1]$ and let $\{[a_\nu,b_\nu]\}_{\nu=1}^{\infty}$ be a sequence of intervals in [0,1] with

$$s \in [a_\nu,b_\nu] \quad \text{and} \quad \lim_{\nu\to\infty}(b_\nu-a_\nu) = 0.$$

For each ν let $\{z_i^\nu\}_{i=1}^{\mu}$ be distinct points in $[a_\nu,b_\nu]$. Suppose $g \in C[0,1]$ and $g_\nu \in C^{\mu-1}[0,1]$ with

$$\max_{[a_\nu,b_\nu]} |g(x) - g_\nu(x)| \to 0 \text{ as } \nu \to 0.$$

Also assume that $\max\limits_{[a_\nu,b_\nu]} |g_\nu^{(\ell)}(x)| \le c,\ 0 \le \ell \le \mu - 1.$

Let $q_\nu \in P_{\mu-1}$ be the Lagrange interpolating polynomial of each g_ν on the points $\{z_i^\nu\}_{i=1}^{\mu}$.

Then

$$\max_{[a_\nu,b_\nu]} |q_\nu(x) - g(x)| \to 0 \text{ as } \nu \to \infty.$$

Proof

For each ν there is a point $\eta_\nu \in [a_\nu,b_\nu]$ such that $q_\nu^{(\mu-1)}(x) \equiv q_\nu^{(\mu-1)}(\eta_\nu) = g_\nu^{(\mu-1)}(\eta_\nu)$.

Thus

$$\max_{[a_\nu,b_\nu]} |q_\nu^{(\mu-1)}(x) - g_\nu^{(\mu-1)}(x)| =$$

$$= \max |q_\nu^{(\mu-1)}(x) - g_\nu^{(\mu-1)}(\eta_\nu) + g_\nu^{(\mu-1)}(\eta_\nu) - g_\nu^{(\mu-1)}(x)|$$

$$= \max |g_\nu^{(\mu-1)}(\eta_\nu) - g_\nu^{(\mu-1)}(x)| \leq 2\ c.$$

It follows that

$$\max |q_\nu(x) - g_\nu(x)| \leq 2\ c(b_\nu - a_\nu)^{\mu-1},$$

and that

$$\max |q_\nu(x) - g(x)| \leq \max |q_\nu(x) - g_\nu(x)| + \max|g_\nu(x) - g(x)|$$

$$\leq 2\ c(b_\nu - a_\nu)^{\mu-1} + \max|g_\nu(x) - g(x)| \to 0 \text{ as } \nu \to \infty. \square$$

<u>Theorem 3.4</u>

Let (3.2 a,b) have the zero solution only. For each j let the collocation points $\{z_{j,i}\}_{i=1}^{m}$ be distinct and contained in $[x_j, x_{j+n}]$. Assume that $f, a_\ell \in C^{m-1}[0,1]$. Then there exist positive contants K and δ, such that (3.3), (3.5) admit a unique solution $\underset{\sim}{p}_h \in P_h^{n,m}$ and such that

$$||\underset{\sim}{p}_h||_{n+m-1} \leq K\{ \max_{0\leq k\leq m-1} ||f^{(k)}||_\infty + \max_{1\leq \ell\leq n} |\beta_\ell|\},$$

whenever $|h| \in (0,\delta]$.

<u>Proof</u>

Again the existence should be established first. As in Theorem 2.2 the proof of the existence part closely follows the proof of the second conclusion of the theorem. Hence only the latter is given below.

If the bound does not hold then one can find sequences $f^\nu \in C^{m-1}[0,1]$, β_ℓ^ν; with $||f^{\nu(\ell)}||_\infty \to 0$, $0 \leq \ell \leq m-1$, and $|\beta_\ell^\nu| \to 0$, $1 \leq \ell \leq n$, as $\nu \to \infty$; and corresponding solutions $\underset{\sim}{p}_{h^\nu} \in P_{h^\nu}^{n,m}$ with $||\underset{\sim}{p}_{h^\nu}||_{n+m-1} = 1$. By Lemma 3.1 there is a subsequence $\{\underset{\sim}{p}_{h^\nu}\}_{\nu=1}^{\infty}$ and a function $p \in C^{n-1}[0,1]$ such that $||\underset{\sim}{p}_{h^\nu} - p||_{n-1} \to 0$ as $\nu \to \infty$. Next reconsider the estimate (3.6):

$$|p_{j_\nu}^{(n)}(s) + \sum_{\ell=0}^{n-1} a_\ell(s)p^{(\ell)}(s)| \le |\sum_{i=1}^{m} \psi_i(s)f^{\nu}(z_i^{\nu})| +$$

$$+ \sum_{\ell=0}^{n-1} |a_\ell(s)p^{(\ell)}(s) - \sum_{i=1}^{m} \psi_i(s)a_\ell(z_i^{\nu})p_{j_\nu}^{(\ell)}(z_i^{\nu})|.$$

Now Lemma 3.3, (with $\mu = m$), applies to each of the n + 1 expressions that appear inside absolute value signs above. Therefore the righthand side of the inequality can be made arbitrarily small by choosing ν sufficiently large. Again this convergence can be shown to be uniform in s. (The details require use of the final inequality in the proof of Lemma 3.3). By an argument similar to that given in the proof of Theorem 2.2 it now follows that $p \in C^n[0,1]$ and that $Lp = 0$. Also p satisfies the homogeneous boundary conditions (3.2 b). However we cannot conclude at this point that $p \not\equiv 0$, unless $m = 1$, since for this we must show first that

$||p_{h^\nu} - p||_{n+m-1} \to 0$ as $\nu \to \infty$. Assume therefore that

$m > 1$. Rewrite the equation $Lp = 0$ as

$$p^{(n)}(x) = - \sum_{\ell=0}^{n-1} a_\ell(x)\, p^{(\ell)}(x).$$

Thus $p^{(n)} \in C^1[0,1]$, i.e. $p \in C^{n+1}[0,1]$, and

$$L^{[1]}p \equiv (Lp)' \equiv p^{(n+1)}(x) + \sum_{\ell=0}^{n} a_\ell^{[1]}(x)p^{(\ell)}(x) = 0.$$

In view of the continuity assumptions on the coefficient functions $a_\ell(x)$ we can repeat this argument another m - 2 times and finally conclude $p \in C^{n+m-1}[0,1]$.

By Rolle's theorem there are points $\xi_i \equiv \xi_{j_\nu,i}^{[1]}$ in $(x_{j_\nu}, x_{j_\nu+n})$ such that $L^{[1]}\, p_{j_\nu}(\xi_i) = f^{\nu'}(\xi_i)$, $1 \le i \le m - 1$.

Let $\psi_i \equiv \psi_{j_\nu,i}^{[1]}$ be the Lagrange interpolating coefficients for the points ξ_i. Then

$$|p_j^{(n+1)}(s) + \sum_{\ell=0}^{n} a_\ell^{[1]}(s)p^{(\ell)}(s)| \le |\sum_{i=1}^{m-1} \psi_i(s)f^{\nu'}(\xi_i)| +$$

$$+ \sum_{\ell=0}^{n} |a_\ell^{[1]}(s)p^{(\ell)}(s) - \sum_{i=1}^{m-1} \psi_i(s)a_\ell^{[1]}(\xi_i)p_{j_\nu}^{(\ell)}(\xi_i)|.$$

With the aid of Lemma 3.3 it follows that the righthand side of the inequality above goes to zero uniformly in s as $\nu \to \infty$. Thus $p_{j_\nu}^{(n+1)} \to p^{(n+1)}$. The above can be repeated another m - 2 times to finally yield

$$\left|\left|\underline{p}_{h^\nu} - p\right|\right|_{n+m-1} \to 0 \text{ as } \nu \to \infty.$$

Since $||\underline{p}_{h^\nu}||_{n+m-1} = 1$ we can conclude that $p \not\equiv 0$, which is a contradiction. □

To derive some error estimates the well known concept of truncation error is required. Let

$$\underline{\rho}_h \equiv \{\rho_j\}_{j=0}^{J-n} \quad \text{where } \rho_j \in P_{n+m-1} \quad \text{interpolates}$$

$$y(x) \text{ at } \{x_{j+i}\}_{i=0}^{n} \cup \{t_{j,i}\}_{i=1.}^{m-1}$$

Here we have introduced points $t_{j,i} \in [x_j, x_{j+n}]$. Thus

$$y(x) - \rho_j(x) = r_j(x) d_j(x), \text{ where}$$

$$r_j(x) \equiv \prod_{i=0}^{n} (x - x_{j+i}) \prod_{i=1}^{m-1} (x - t_{j,i}) \quad \text{and where } d_j(x)$$

denotes the appropriate divided difference. Define

$$\underline{\tau}_h \equiv \{\tau_j\}_{j=0}^{J-n} \quad \text{with} \quad \tau_j(x) \equiv Lp_j(x) - L\rho_j(x).$$

The local truncation errors can now be defined as the values of $\tau_j(x)$ at $x = z_{j,i}$. We have

$$\tau_j(z_{j,i}) = f(z_{j,i}) - L\rho_j(z_{j,i}) = L(y-\rho_j)(z_{j,i}) = L(r_j d_j)(z_{j,i}).$$

If y(x) is sufficiently differentiable then $\tau_j(z_{j,i}) = O(|h|^m)$ regardless of the choice of collocation points $z_{j,i}$. For certain special choices of the $z_{j,i}$ this estimate can be improved. Certain of these special points can be found by evaluating $L(r_j d_j)(z_{j,i})$ and equating the leading terms to zero. Examples of this procedure are given in Doedel (1978,1979).

Theorem 3.5

Let the homogeneous problem (3.2 a,b) only admit the zero solution. Let $a_\ell \in C[0,1]$, $y \in C^{n+m}[0,1]$ and assume that $z_{j,i} \in [x_j, x_{j+n}]$, with

$$\min_{i_1 \neq i_2} |z_{j,i_1} - z_{j,i_2}| \geq c_1 |x_{j+n} - x_j|.$$ Also assume that

$$\max_{i,j} |\tau_j(z_{j,i})| \leq c_2 |h|^{m+\sigma}$$ for some nonnegative integer σ, and that in addition

$$|B_\ell \underline{\rho}_h - \beta_\ell| \leq c_3 |h|^{m+\mu}, \quad 1 \leq k \leq n, \ \mu \geq \sigma .$$

Here c_1, c_2 and c_3 are constants that do not depend on j and h, when $|h|$ is small enough. Then there exist positive constants C and δ such that

$$\max_j \max_{[x_j, x_{j+n}]} |p_j^{(\ell)}(x) - y^{(\ell)}(x)| \leq c|h|^{\min\{m+\sigma,\ n+m-\ell\}}, \quad 0 \leq \ell < n,$$

whenever $|h| \in (0,\delta]$.

Proof

$$|y^{(\ell)}(x) - p_j^{(\ell)}(x)| \leq |y^{(\ell)}(x) - \rho_j^{(\ell)}(x)| + |\rho_j^{(\ell)}(x) - p_j^{(\ell)}(x)|$$

The first term on the righthand side of this inequality is a local interpolation error. The second term can be estimated with Theorem 3.2. We have

$$\max_j \max_{[x_j, x_{j+n}]} | y^{(\ell)}(x) - p_j^{(\ell)}(x)|$$

$$\leq c_4 |h|^{n+m-\ell} + K\{\max_{i,j} |\tau_j(z_{j,i})| + \max_\ell | B_\ell \underline{\rho}_h - \beta_\ell|\}$$

$$\leq c_4 |h|^{n+m-\ell} + K(c_2+c_3)|h|^{m+\sigma} . \square$$

Acknowledgement

This research was supported by U.S. Army Grant #DAAG29-78-G-0126.

REFERENCES

E. J. Doedel, The construction of finite difference approximations to ordinary differential equations, SIAM J. Num. Anal., Vol. 15, No. 3, 1978, pp. 450-465.

E. J. Doedel, Finite difference collocation methods for nonlinear two point boundary value problems, SIAM J. Num. Anal., Vol. 16, No. 2, 1979.

H. B. Keller, Approximation methods for nonlinear problems with application to two point boundary value problems, Math. of Comp. 29, 1975, pp. 464-474.

H. O. Kreiss, Difference approximations for boundary and eigenvalue problems for ordinary differential equations, Math. of Comp. 26, 1972, pp. 605-624.

R. E. Lynch and J. R. Rice, The HODIE method, Report CSD-TR 170, Department of Computer Science, Purdue University, 1976.

M. R. Osborne, A method for finite difference approximation to ordinary differential equations, The Computer Journal, Vol. 7, No. 1, 1964, pp. 58-65.

M. R. Osborne, Collocation, difference equations, and stitched function representations, Proc. Dublin Conf. in Numerical Analysis, 1974.

V. Pereyra, Iterated deferred corrections for nonlinear operator equations, Numer. Math. 10, 1967, pp. 316-323.

G. W. Reddien, Survey on Projection Methods, to appear in SIAM Review.

FAST APPROXIMATIONS FOR RICCATI EQUATIONS VIA WALSH FUNCTIONS AND BLOCK PULSE FUNCTIONS

Oren N. Dalton
Mathematical Services Branch
Analysis and Computation Division
National Range Operations Directorate
White Sands Missile Range, New Mexico 88002

ABSTRACT. Recent investigations into the use of Walsh function approximation for filtering and control problems demonstrate that such functions can provide satisfactory approximations for linear and nonlinear systems, and often substantially decrease the computational burden. Precision is limited, of course, by the number, m, of Walsh functions used in an expansion. This number, in turn, was thought to be limited because the solution appeared to demand a number of matrix inversions proportional to m. In this paper the author presents a background problem and proves that only a single matrix inverse is required, independent of m. Moreover, it is shown that a problem formulated in Walsh functions has a solution, inevitably, in block pulse functions. References are cited showing computational results.

1. PROBLEM OUTLINE.

The matrix Riccati equations for optimal control and filtering have appeared in the literature many times since Kalman first introduced them [1]. They will merely be listed here for reference:

$$\dot{x} = Xx + Uu$$

$$z = Hx + v$$

where x is the state vector for system dynamics, u a stocastic input or control vector, z an observation vector, and v a zero-mean vector of noise. The usual conditions of non-correlation across time of u and v, and non-correlation of u with v are assumed. The symbols Q and R, as functions of time, represent the covariance matrices of u and v respectively. If, now $\hat{x}$ and $\dot{\hat{x}}$ represent the optimal estimates of the state vector we also have [1]:

$$\dot{\hat{x}} = X\hat{x} + K[z - H\hat{x}]$$

$$K = PH^T R^{-1}$$

$$\dot{P} = UQU^T + XP + PX^T - PH^T R^{-1} H P$$

where K is the Kalman gain and P is the covariance matrix of the state vector.

We will address the problem of estimating P.

For ease of notation, define:

$$A \triangleq UQU^T$$

$$B \triangleq H^T R^{-1} H$$

Then $\dot{P}$ can be written as:

$$\dot{P} = A + XP + PX^T - PBP$$

We assume that P is absolutely integrable over any interval of interest and is full rank. With these considerations, define differentiable vectors a and b as:

$$a = Pb$$

$$\dot{a} = \dot{P}b + P\dot{b}$$

and multiplying $\dot{P}$ on the right by b we have:

$$\dot{P}b = Ab + XPb + PX^T b - PBPb$$

$$= \dot{a} - P\dot{b} = Ab + Xa + PX^T b - PBa$$

Since the solution for P is unique, we can equate like powers of P on each side of the last two terms to obtain the simultaneous vector equations:

$$\dot{a} = Ab + Xa$$

$$\dot{b} = -X^T b + Ba$$

which, when solved, can be used to construct P. Create the vector V and matrix M as:

$$V \triangleq \begin{bmatrix} a \\ b \end{bmatrix}, \quad M \triangleq \begin{bmatrix} X & A \\ B & -X^T \end{bmatrix}$$

and the system to be solved can be written as:

$$\dot{V} = MV$$

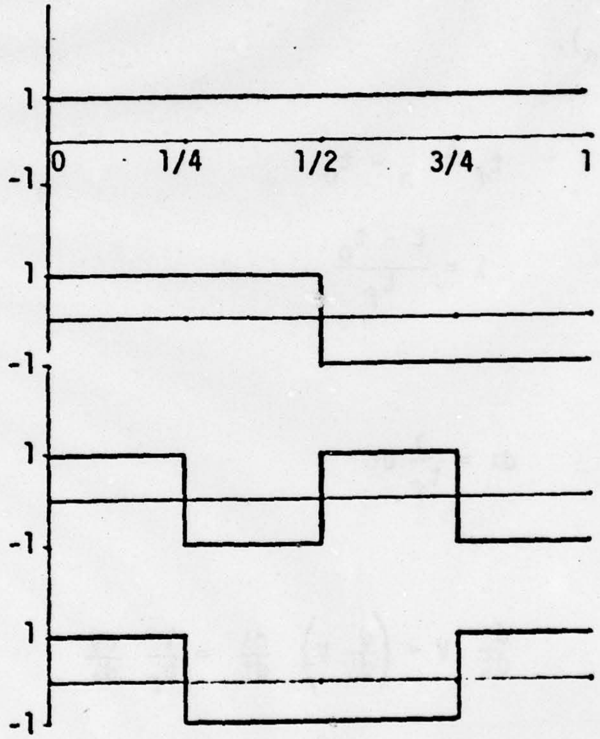
From the above development, it is clear that the resultant equation can be applied to problems more general than just the one implied. The main import of this paper will be to approximate V by means of Walsh functions, and in particular, to use to advantage the structure of the formulation to produce a fast approximating algorithm.

2. A NOTE ABOUT WALSH FUNCTIONS.

In the last section, it was pointed out that we wish to approximate the solution to the vector Riccati equation

$$\dot{V} = HV$$

in terms of Walsh functions [2]. Walsh functions are square waves defined over the interval [0,1] which assume the values of $\pm$ 1 for varying lengths in the interval. Harmuth [3] illustrates the first sixty-four Walsh functions, and lesser numbers appear in [4, 5, 6]. The first four are shown below:



A convenient way to represent these functions is by means of the Walsh matrix in which the $i^{\underline{th}}$ row corresponds to the $i^{\underline{th}}$ Walsh function, and the entries in the columns correspond to the value of the function over the specified subinterval. For the example in the above figure, W is:

$$W = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

We note that $W = W^T$ and $W^{-1} = (1/m)W$, where m is the order of W.

3. EXPANSION IN TERMS OF WALSH FUNCTIONS.

Assume that the vector, V, is defined over the interval $[t_o, t_n)$, where $\dot{V} \triangleq dV/dt$, $t\varepsilon[t_o, t_n)$.

Let

$$t_f \triangleq t_n - t_o$$

$$\lambda = \frac{t - t_o}{t_f}$$

then

$$d\lambda = \frac{1}{t_f} dt$$

and

$$\frac{d}{dt} V = \left(\frac{d}{d\lambda} V\right) \frac{d\lambda}{dt} = \frac{1}{t_f} \frac{dV}{d\lambda}$$

If we now let the dot notation imply differentiation with respect to $\lambda, \lambda\varepsilon[0,1)$, we have:

$$\dot{V} = t_f MV$$

and assuming that t_f is absorbed into M, we can derive an approximation, without loss of generality, for the equation:

$$\dot{V} = MV, \lambda\varepsilon[0,1)$$

Chen and Hsaio [4, 5, 6] show that the integral of a Walsh function can be approximated by an expansion in terms of Walsh functions. Let $\phi_m(\lambda)$ be a vector of m Walsh functions. Then

$$\int_0^\lambda \phi_m(s)ds = P_m\phi_m(\lambda)$$

where P_m is the integration operator matrix.

To formulate the problem, here, we proceed, as do Chen and Hsaio [4] as follows: Expand V in terms of Walsh functions:

$$\dot{V} = C\phi_m$$

where C is a matrix of coefficients. Then

$$\int_{V_o}^{V(\lambda)} dV = V(\lambda) - V_o = \int_0^\lambda C\phi_m(s)\, ds = C\, P_m\, \phi_m(\lambda)$$

or

$$V = CP_m\phi_m + V_o$$

From the equation

$$\dot{V} = MV$$

we have:

$$C\phi_m = M(CP_m\phi_m + V_o)$$

Now, since the first Walsh function is identically one for all $\lambda\epsilon[0,1)$, if we form the matrix

$$V^* \triangleq [MV_o \quad 0]$$

where only the first column of V* is non-zero, the above becomes:

$$C\phi_m = (MCP_m + V^*)\phi_m$$

from which

$$C = MCP_m + V^*$$

At this point, we introduce the two operators: vec (·) and $\otimes$. For example see [7, 8]. The first operator vec, converts a matrix into a vector of its columns. Representing C by its columns as:

$$C = [C_1\ C_2\ \ldots C_m]$$

define c and k using the vec operators as

$$c \triangleq \text{vec}\ (C) = \begin{bmatrix} C_1 \\ C_2 \\ \cdot \\ \cdot \\ \cdot \\ C_m \end{bmatrix}$$

and

$$k \triangleq \text{vec}\ (V^*) \quad \begin{bmatrix} MV_o \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix}$$

The second operator represented by the symbol, $\otimes$, is the Kronecker product; it is defined, here, left-to-right (some authors define it oppositely) between two matrices, say A and B as

$$A \otimes B = \begin{bmatrix} b_{11}A & b_{12}A & \cdots \\ b_{21}A & \cdot & \\ \cdot & \cdot & \\ \cdot & \ddots & \cdot \\ \cdot & & \vdots \end{bmatrix}$$

It can be shown [8] that

$$\mathrm{Vec}\ (MCP_m) = M \otimes P_m^T\ \mathrm{vec}\ (C)$$

Thus we have:

$$c = M \otimes P_m^T c + k$$

from which

$$c = (I - M \otimes P_m^T)^{-1} k$$

It is clear that the major problem is to invert the matrix

$$I - M \otimes P_m^T$$

If the number of elements in the state vector is n, this matrix is of size (2nm x 2nm). Reasonable requirements of precision would suggest that the number of Walsh functions retained in the expansion may be quite large, so that a matrix of order 2nm would become unwieldy very quickly. A number of schemes have been proposed for this inversion, taking advantage of the structure of the Walsh matrix to produce recursive algorithms. Chen and Hsiao [5], for example, recursively invert a number of matrices, depending on m, of order 2n (for the present problem). Stavroulakis and Tzafestas [9] chose an interval size so that a resultant number, m_1, is sufficiently large so that the following approximation is valid.

$$(I + \frac{1}{m_1} A)^{-1} \approx I - \frac{1}{m_1} A$$

They then recursively compute the required vectors.

In this paper it will be shown that it is necessary to invert only one 2n x 2n matrix (or two n x n matrices) irrespective of the value of m or the size of the interval.

Chen and Hsaio have shown [4, 5] that if the number of Walsh functions, m, is

$$m = 2^{\alpha},\ \alpha \varepsilon \mathbb{N}\ ,\ \text{the natural numbers,}$$

then the integration operator P_m, has a particularly simple form, namely:

$$P_m = \begin{bmatrix} P_{m/2} & -\frac{1}{2m} I_{m/2} \\ \frac{1}{2m} I_{m/2} & O_{m/2} \end{bmatrix}$$

where I_k and 0_k represent a unit matrix and a matrix of zeros of order k, respectively. This matrix is illustrated in expanded detail and specifically for m = 16 in [5]. It is also illustrated up to m = 8 in [4, 6, 9].

For this paper we redefine P_m slightly; this makes subsequent work somewhat easier. We note [10] that P_m can be written as:

$$P_m = \frac{1}{2m}\begin{bmatrix} 2mP_{m/2} & -I_{m/2} \\ I_{m/2} & 0_{m/2} \end{bmatrix} \triangleq \frac{1}{2m}\ P^*_m$$

From the definition of P^*_m

$$2mP_{m/2} = \frac{2m}{m}\ P^*_{m/2}$$

so we can write:

$$P_m = \frac{1}{2m}\begin{bmatrix} 2P^*_{m/2} & -\ I_{m/2} \\ I_{m/2} & 0_{m/2} \end{bmatrix}.$$

This definition also allows us define $P^*_1 = [1]$.

The starred form of the integration matrix will be used through out the rest of this paper so the "*" will be dropped. This necessitates a slight readjustment of the equations for c and V to:

$$c = (I - \frac{1}{2m}\ M \otimes P_m^T)^{-1}k$$

$$= 2m(2mI - M \otimes P_m^T)^{-1}\ k$$

$$V = \frac{1}{2m}\ CP_m\phi_m + V_0\ .$$

Now the major problem is to invert the matrix we will define as Γ_2:

$$\Gamma_2 \triangleq 2mI - M \otimes P_m^T$$

Gopalsami and Deekshatula [11] pointed out that P_m^T could be transformed to a particularly simple form using the Walsh matrix W, as follows:

$$W P_m^T W^{-1} = \begin{bmatrix} 1 & 0 & 0 & & & & \\ 2 & 1 & 0 & & 0 & & \\ 2 & 2 & 1 & & & & \\ & \cdot & & \cdot & & & \\ & \cdot & & & \cdot & & \\ & \cdot & & & & \cdot & \\ 2 & 2 & 2 & \cdot & \cdot & \cdot & 1 \end{bmatrix} \triangleq \Sigma^T$$

Or, since

$$W^{-1} = \frac{1}{m} W$$

$$W P_m^T W = m \Sigma^T$$

Define:

$$\Gamma_1 = (I \otimes W) \Gamma_2 (I \otimes W)^{-1}$$

which implies:

$$\Gamma_1^{-1} = (I \otimes W) \Gamma_2^{-1} (I \otimes W)^{-1}$$

from which

$$\Gamma_2^{-1} = (I \otimes W)^{-1} \Gamma_1^{-1} (I \otimes W)$$

This can be rewritten [7, 8] as:

$$\Gamma_2^{-1} = (I \otimes W^{-1}) \Gamma_1^{-1} (I \otimes W) = \frac{1}{m} (I \otimes W) \Gamma_1^{-1} (I \otimes W)$$

Observing the detail of Γ_1:

$$\Gamma_1 = (I \otimes W) \Gamma_2 (I \otimes W)^{-1}$$

$$= (I \otimes W)(2mI - M \otimes P_m^T)(I \otimes W)^{-1}$$

Γ_1 can be solved [7, 8] as:

$$\Gamma_1 = 2m(I \otimes W)(I \otimes W)^{-1} - (I\cdot M) \otimes (W\cdot P_m^T)(I \otimes W^{-1})$$

$$= 2mI - M \otimes (W\, P_m^T W^{-1})$$

$$= 2mI - M \otimes \sum^T$$

Make the following definitions:

$$F \triangleq 2mI - M$$

$$G \triangleq 2F^{-1}\, M$$

$$E \triangleq I + G$$

Then, using the definition of Σ^T, Γ_1 has the appearance:

$$\Gamma_1 = \begin{bmatrix} F & 0 & \cdot & \\ -2M & F & & 0 \\ -2M & -2M & \cdot & \\ \cdot & \cdot & \cdot & \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \\ -2M & -2M & \cdot\ \cdot\ \cdot & F \end{bmatrix}$$

and it is easy to verify (or prove by induction) that

$$\Gamma_1^{-1} = \begin{bmatrix} I & 0 & & \\ G & I & & 0 \\ GE & G & & \\ \cdot & & \cdot & \\ \cdot & \cdot & \cdot & \\ \cdot & \cdot & & \cdot \\ GE^{m-2} & GE^{m-3} & & I \end{bmatrix} (F^{-1} \otimes I)$$

In the term for Γ_1^{-1} define the lower triangular matrix as S; that is:

$$\Gamma_1^{-1} = S(F^{-1} \otimes I)$$

Thus

$$\Gamma_2^{-1} = \frac{1}{m}(I \otimes W)S(F^{-1} \otimes I)(I \otimes W)$$

and

$$c = 2m\,\Gamma_2^{-1}k = 2\,(I \otimes W)\,S\,(F^{-1} \otimes I)(I \otimes W)\,k$$

It is easy to see that

$$(F^{-1} \otimes I)(I \otimes W) = (I \otimes W)\,(F^{-1} \otimes I)$$

since only F^{-1} and unit matrices are involved. Thus:

$$c = 2(I \otimes W)\,S(I \otimes W)(F^{-1} \otimes I)\,k$$

Now by definition,

$$k = \begin{bmatrix} MV_0 \\ 0 \\ 0 \\ \cdot \\ \vdots \\ \cdot \\ 0 \end{bmatrix}$$

so

$$(F^{-1} \otimes I)k = \begin{bmatrix} F^{-1}MV_0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} GV_0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix}$$

Moreover, since the first block column of $(I \otimes W)$ consists of positive unit matrices, we have that:

$$(I \otimes W)(F^{-1} \otimes I)\, k = \frac{1}{2} \begin{bmatrix} GV_o \\ GV_o \\ \cdot \\ \cdot \\ \cdot \\ GV_o \end{bmatrix} \triangleq \frac{1}{2}\, y$$

Thus c becomes:

$$c = (I \otimes W)\, S\, y$$

From the form of S, and factoring GV_o, each term in Sy, with the exception of the first, is of the form

$$(I + G \sum_{i=0}^{k} E^i) GV_o \text{ , } k = 0, 1, \ldots, m-2$$

which equals $E^{k+1} GV_o$. This is easily seen from

$$I + G \sum_{i=0}^{k} E^i = I + (E-I) \sum_{i=0}^{k} E^i$$

$$= I + \sum_{i=1}^{k+1} E^i - \sum_{i=0}^{k} E^i$$

$$= E^{k+1}$$

Notice that this equation has tacitly assumed that $m \geq 2$; the case for m = 1 is trivial and will be ignored. Thus, the vector Sy is:

$$Sy = \begin{bmatrix} GV_o \\ EGV_o \\ E^2GV_o \\ \cdot \\ \cdot \\ \cdot \\ E^{m-1}GV_o \end{bmatrix}$$

If we now define the matrix

$$Y \triangleq [GV_o \; EGV_o \; \ldots \; E^{m-1}GV_o]$$

then the inverse of the vec operator (based on vectors of length 2n) results in:

$$vec^{-1}\,(c) = C = vec^{-1}\,(I \otimes W)Sy \quad = YW$$

and the solutions for V and $\dot{V}$ become:

$$\dot{V} = YW\phi_m \triangleq mY\psi_m$$

$$V = \frac{1}{2m}\, YW\, P_m\phi_m + V_o$$

$$= \frac{1}{2m}\, YW\, P_m W^{-1} W\phi_m + V_o$$

$$\triangleq \frac{1}{2m}\, Y\Sigma(m\psi_m) + V_o$$

$$= \frac{1}{2}\, Y\Sigma\; \psi_m + V_o$$

Here ψ_m is a vector of a subset of m orthogonal block pulse functions (shown below for m = 4). It is known [10, 11] that the product of the Walsh matrix, W, and a vector of Walsh functions is

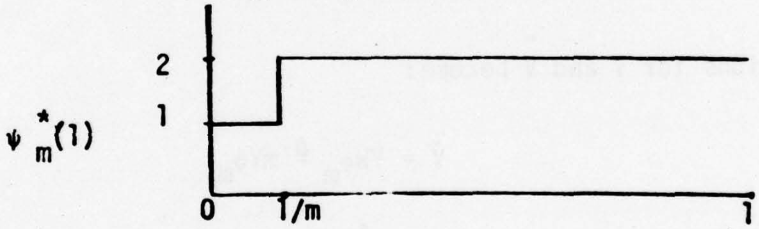
$$m\psi_m = W\phi_m$$

As the writer pointed out in [10], the above result is rather surprising. The problem was formulated as an expansion in Walsh functions but the solution is in terms of an orthonormal subset of block pulse functions, whether we like it or not. Note that Σ is the integration matrix for the block pulses.

The set of functions, say $\psi_m^* \triangleq \Sigma\psi_m$ are interesting in their own right. Now, Σ is the form:

$$\Sigma = \begin{bmatrix} 1 & 2 & 2 & \cdots & 2 \\ 0 & 1 & 2 & \cdots & 2 \\ & & \cdot & & \\ & 0 & & \cdot & \\ & & & & \cdot \\ & & & & 1 \end{bmatrix}$$

and referring to the figure below, when $\lambda \varepsilon [0,1/m)$ the first block pulse is unity and the rest zero. When $\lambda \varepsilon [1/m, 2/m)$ the second one is unity and the rest are zero. Thus, the first term in $\psi_m^* = \Sigma\psi_m$ over the interval [0,1) is:



From similar reasoning, each term in ψ_m^* is of the same shape, but the $i^{\underline{th}}$ term begins at (i -1)/m. Except for a multiplicative constant, the elements in ψ_m^* are the averages of the integrals of the corresponding elements [10] in ψ_m. A sketch of ψ_4, $\int \psi_4$ and ψ_4^* is shown on the next page.

The solution of V is now:

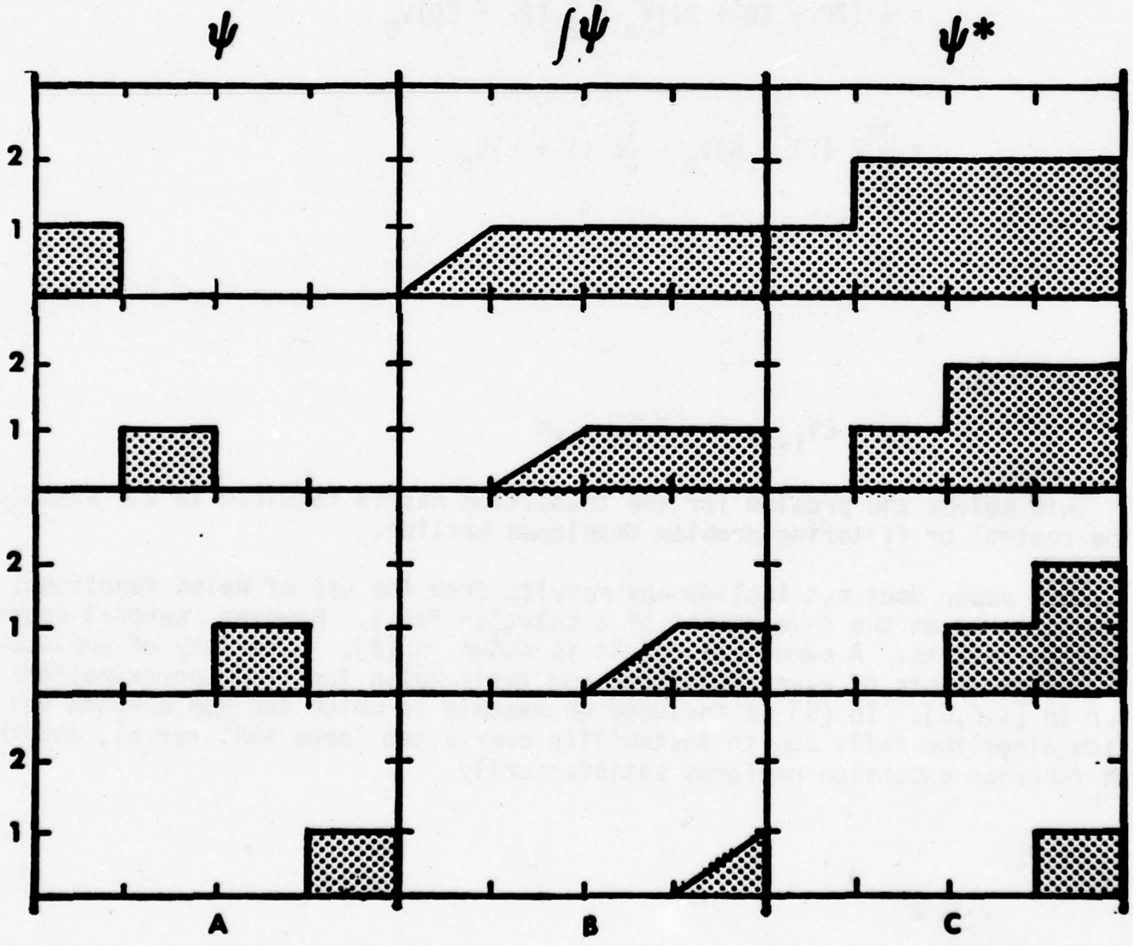
$$V(\lambda) = \frac{1}{2} Y\psi_m^*(\lambda) + V_o$$

When $\psi\varepsilon[0,1/m)$, according to the sketch below, the first term in ψ_m^* is unity and the rest are zero. Distinguish the first interval by the subscript "1", the next by the subscript "2", and so on. Then, from the definition

E = I + G, given earlier:

$$V_1 = \frac{1}{2} Y \begin{bmatrix} 1 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix} + V_o = \frac{1}{2} GV_o + V_o = \frac{1}{2} (G + 2I)V_o$$

$$= \frac{1}{2} (I + E)V_o$$



The first four (A) block pulse functions, (B) block pulse function integrals, and (C) the functions $\psi^* = \Sigma\psi$.

In the second subinterval, the first term in ψ_m^* is two, the second, one, and the rest zero. Thus:

$$V_2 = \frac{1}{2} Y \begin{bmatrix} 2 \\ 1 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix} + V_o = \frac{1}{2} (2GV_o + EGV_o) + V_o$$

$$= \frac{1}{2} (2G + EG + 2I)V_o = \frac{1}{2} (2E + EG)V_o$$

$$= \frac{1}{2} E (2I + G)V_o = \frac{1}{2} E (I + E)V_o$$

$$= EV_1$$

and in general:

$$V_i = EV_{i-1}, \; i = 2,3,\ldots,m$$

This solves the problem for the transition matrix required in the solution of the control or filtering problem developed earlier.

This paper does not include any results from the use of Walsh functions; its emphasis is on the development of a solution for V. However, several sources do include results. A numerical result is shown in [9]. A variety of applications showing plots of exact solutions and their Walsh function approximations appear in [4,5,6]. In [5] is included an example in which the Runga-Kutta integration algorithm fails due to instability over a too large subinterval, and the Walsh function expansion performs satisfactorily.

5. REFERENCES.

1. Kalman, R.E., "A New Approach to Linear Filtering and Prediction Problems", J. of Basic Engineering, Trans. of the ASME, Mar. 1960, pp. 35-44.

2. Walsh, J.L., "A Closed Set of Orthogonal Functions", Am. J. of Math., (45), 1923, pp. 2-24.

3. Harmuth, H.F., "Transmission of Information by Orthogonal Functions", Springer-Verlag, N.Y., 1972, 2nd ed., p. 5.

4. Chen, C.F. and Hsiao, C.H., "Design of Piecewise Constant Gains for Optimal Control via Walsh Functions", IEEE Trans. on Aut. Cont. (AC-20)5, Oct 1975, 596-603.

5. ______, "A State-space Approach to Walsh Series Solutions of Linear Systems", Int. J. of Sys. Sci., (6)9, 1975, pp. 833-58.

6. ______, "A Walsh Series Direct Method for Solving Variational Problems", J. of the Franklin Inst., (300)4, Oct. 1975, pp. 265-80.

7. Neudecker, H., SIAM J. Appl. Math., (17), 1969, p. 667.

8. Powers, D.L., "Solving AX + XB = C by Control and Tearing", Int. J. Cont., (23)3, 1976, p. 421.

9. Stavroulakis, P. and Tzafestas, S., "Walsh Series Approach to Observer and Filter Design in Optimal Control Systems", Int. J. Cont., (26)5, 1977, pp 721-6.

10. Dalton, O.N., "Further Comments on Design of Piecewise Constant Gains for Optimal Control via Walsh Functions", IEEE Trans. on Aut. Cont., (AC-23)4, Aug. 1978, 760-3.

11. Gopalsami, N. and Deekshatulu, B.L., "Comments on 'Design of Piecewise Constant Gains for Optimal Control via Walsh Functions'", IEEE Trans. on Aut. Cont., (AC-21)4, Aug 1976, pp. 634-6.

THE CONSTRUCTION OF JACOBI AND PERIODIC JACOBI MATRICES

WITH PRESCRIBED SPECTRA

Warren E. Ferguson, Jr.
Mathematics Research Center
University of Wisconsin - Madison
610 Walnut Street
Madison, Wisconsin 53706

and

Mathematics Department #89
University of Arizona
Tucson, Arizona 85721

ABSTRACT. The spectral properties of Jacobi and periodic Jacobi matrices are analyzed and algorithms for the construction of Jacobi and periodic Jacobi matrices with prescribed spectra are presented. Numerical evidence demonstrates that these algorithms are of practical utility. These algorithms have been used in studies of the periodic Toda lattice, and might also be used in studies of inverse eigenvalue problems for Sturm-Liouville equations and Hill's equation.

1. Introduction. A periodic Jacobi matrix is any real, symmetric matrix of the form

$$L = \begin{bmatrix} a_1 & b_1 & & & b_N \\ b_1 & \ddots & \ddots & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & \ddots & \ddots & b_{N-1} \\ b_N & & & b_{N-1} & a_N \end{bmatrix} \qquad \text{where} \quad b_i > 0 \quad \forall\, i .$$

This paper shows how one can construct a periodic Jacobi matrix with prescribed spectra. For example, there is a family of periodic Jacobi matrices with $\lambda_1, \cdots, \lambda_N$ as eigenvalues if and only if the numbers $\lambda_1, \cdots, \lambda_N$ are real and can be ordered so that

$$\lambda_1 > \lambda_2 \geq \lambda_3 > \lambda_4 \geq \lambda_5 > \cdots .$$

Similar problems have been studied by other authors [2, 12].

The results presented in this paper are based upon an analysis of the spectral properties of periodic Jacobi matrices. The main tool in this analysis is the knowledge of the spectral properties of Jacobi matrices. Recall that a Jacobi matrix is any real, symmetric tridiagonal matrix whose next to diagonal entries are positive. Our cannonical Jacobi matrix will be the matrix obtained by deleting from L the last row and column, that is

Sponsored by the United States Army under Contract No. DAAG29-75-C-0024. This material is based upon work supported by the National Science Foundation under Grant No. MCS78-09525.

$$J = \begin{bmatrix} a_1 & b_1 & & 0 \\ b_1 & \ddots & \ddots & \\ & \ddots & \ddots & b_{N-2} \\ 0 & & b_{N-2} & a_{N-1} \end{bmatrix} \qquad \text{where } b_i > 0 \;\; \forall \;\; i.$$

An algorithm which constructs a Jacobi matrix with prescribed spectra is presented in Theorem 2. This algorithm is derived from the fact that any real, symmetric matrix has real eigenvalues and a corresponding full set of real, orthonormal eigenvectors. We hasten to point out that essentially the same algorithm was presented by de Boor and Golub [3]. Similar problems have been studied by other authors [8, 9].

The spectral properties of periodic Jacobi matrices are considered in Section 3. The results presented in this section are derived from a matrix analog of Floquet theory [11]. In Section 4 we use these results to characterize the family of periodic Jacobi matrices with prescribed spectra.

We present the results of several numerical experiments in Section 5. These results demonstrate that the algorithms presented in Theorems 2 and 6 are of practical utility. Indeed, these algorithms have been used in performing numerical experiments on the periodic Toda lattice [4]. In Section 6 we conclude the paper with several comments.

2. Spectral Properties of Jacobi Matrices. In this section we will consider the spectral properties of the Jacobi matrix

$$J = \begin{bmatrix} a_1 & b_1 & & 0 \\ b_1 & \ddots & \ddots & \\ & \ddots & \ddots & b_{N-2} \\ 0 & & b_{N-2} & a_{N-1} \end{bmatrix} \qquad \text{where } b_i > 0 \quad \forall \; i .$$

Observe that J is a real, symmetric matrix. Consequently J has real eigenvalues $\mu_1, \cdots, \mu_{N-1}$ and a corresponding set $Y_1, \cdots, Y_{N-1}$ of real, orthonormal eigenvectors [13,14]. If Y denotes the matrix whose $j\underline{th}$ column is Y_j then Y is an orthogonal matrix and

$$JY = YD \qquad \text{where} \qquad D = \begin{bmatrix} \mu_1 & & 0 \\ & \ddots & \\ 0 & & \mu_{N-1} \end{bmatrix} . \tag{1}$$

Many important relationships between the eigenvalues and eigenvectors of J can be derived from the representation

$$(\mu I - J)^{-1} = Y(\mu I-D)^{-1} Y^T \tag{2}$$

of the resolvent of J . For example, by comparing the entries in row 1, column N-1 of (2) we arrive at the identity

$$b_1 \cdots b_{N-2} = \sum_{k}{}_{1}^{N-1} \frac{\omega(\mu)}{\mu-\mu_k} Y_{1,k} Y_{N-1,k} \ . \tag{3.a}$$

Here

$$\omega(\mu) = \det\ (\mu I-J) \tag{3.b}$$

is the characteristic polynomial of J and $Y_{i,j}$ denotes the entry of Y in row i , column j. Another important identity, used by Stieltjes in his treatment of inverse eigenvalue problems, can be derived from (2) by comparing the entries in row 1, column 1 (or row N-1, column N-1.)

In the work that follows we will demonstrate that the entries of J can be recovered from the entries on the diagonal of D and in the first row of Y . Before we describe this process let us introduce the following:

Definition 1: (a) The Jacobi matrix J is characterized by the data $\{\mu,y\}$ if and only if

(1) $\mu_1,\cdots,\mu_{N-1}$ are the eigenvalues of J , and

(2) $y_1,\cdots,y_{N-1}$ are the first components of a set $Y_1,\cdots,Y_{N-1}$ of real, orthonormal eigenvectors of J corresponding to $\mu_1,\cdots,\mu_{N-1}$.

(b) The data $\{\mu,y\}$ is compatible if and only if

(1) $\mu_1,\cdots,\mu_{N-1}$ are real, distinct numbers, and

(2) $y_1,\cdots,y_{N-1}$ are real, nonzero numbers whose squares sum to one.

We feel justified in using the words "characterize" and "compatible" in this manner because the following theorem is true.

Theorem 2: Data characterizing a Jacobi matrix is compatible. Furthermore each set of compatible data $\{\mu,y\}$ characterizes a unique Jacobi matrix J . The entries (a,b) of this Jacobi matrix are computed by the algorithm:

```
1.  $Y_{0,j} = 0 \quad \forall j$
2.  $Y_{1,j} = y_j \quad \forall j$
3.  For $i = 1,\cdots,N-2$
4.  $a_i = \sum_{k}{}_{1}^{N-1} \mu_k Y_{i,k}^2$
5.  $b_i^2 = \sum_{k}{}_{1}^{N-1} [(\mu_k - a_i)Y_{i,k} - b_{i-1}Y_{i-1,k}]^2$
6.  $Y_{i+1,j} = \frac{1}{b_i}[(\mu_j - a_i)Y_{i,j} - b_{i-1}Y_{i-1,j}] \quad \forall j$
7.  Next $i$
8.  $a_{N-1} = \sum_{k}{}_{1}^{N-1} \mu_k Y_{N-1,k}^2$ .
```

Proof: The proof of this theorem will be presented as a sequence of three lemmas. ∎

Lemma 2.1: Data characterizing a Jacobi matrix is compatible.

Proof: Let the Jacobi matrix J be characterized by the data $\{\mu,y\}$. The μ's are necessarily real because they are the eigenvalues of a real, symmetric matrix. By definition the y's are real, and their squares sum to one because they may be considered to be the entries in the first row of the orthogonal matrix Y in (1). Consider the limiting form of the identity (3) as μ tends to μ_j. If μ_j were a repeated eigenvalue then $\omega'(\mu_j) = 0$ and so we would be forced to conclude that $b_1 \cdots b_{N-2} = 0$, which is impossible because each $b_i > 0$. Therefore the μ's are distinct and, as μ tends to μ_j, we infer that

$$b_1 \cdots b_{N-2} = \omega'(\mu_j)\, Y_{1,j} Y_{N-1,j} \quad \forall j . \tag{4}$$

Consequently the y's are nonzero because $y_j = Y_{1,j}$. ∎

Lemma 2.2: Given compatible data $\{\mu,y\}$ the algorithm of Theorem 2 computes the entries (a,b) of a Jacobi matrix J characterized by the data $\{\mu,y\}$.

Proof: First, we infer that this algorithm computes the entries (a,b) of some Jacobi matrix J only if the value of b_i computed in step 5 is never zero. From the compatibility of the data we infer that $b_1 > 0$. If $b_1, \cdots, b_{\ell-1} > 0$ but $b_\ell = 0$ for some $\ell < N-1$ then step 6 implies

$$\begin{bmatrix} a_1 & b_1 & & 0 \\ b_1 & \ddots & \ddots & \\ & \ddots & \ddots & b_{\ell-1} \\ 0 & & b_{\ell-1} & a_\ell \end{bmatrix} \begin{bmatrix} Y_{i,j} \\ | \\ Y_{\ell,j} \end{bmatrix} = \mu_j \begin{bmatrix} Y_{i,j} \\ | \\ Y_{\ell,j} \end{bmatrix} \qquad \forall j .$$

But this is impossible, for no matrix of order $\ell < N-1$ has $N-1$ distinct eigenvalues.

Second, we will demonstrate that the numbers $Y_{i,j}$ computed by this algorithm form the entries of an orthogonal matrix Y, that is the rows of Y satisfy the orthonormality relations

$$\sum_{k}^{N-1} Y_{i,k} Y_{j,k} = \delta_{i,j} \qquad \text{for} \qquad j = 1,\cdots,i \tag{5}$$

and $i = 1,\cdots,N-1$. From the compatibility of the data $\{\mu,y\}$ we infer that (5) is true for $i = 1$. If (5) is true for $i = 1,\cdots,\ell$ then the following argument demonstrates that it is also true for $i = \ell+1$. Clearly steps 5 and 6 imply that (5) is true for $j = \ell+1$. For $j \leq \ell$ step 6 implies that

$$\sum_{k}^{N-1} Y_{\ell+1,k} Y_{j,k} = \frac{1}{b_\ell} \left[\sum_{k}^{N-1} \mu_k Y_{\ell,k} Y_{j,k} - a_\ell \delta_{\ell,y} - b_{\ell-1} \delta_{\ell-1,j} \right] .$$

The right side of this equality is zero for $j = \ell$ because step 4 was executed, and it is zero for $j < \ell$ because step 6 implies that

$$\sum_{k}^{N-1}{}_{1} \mu_k Y_{\ell,k} Y_{j,k} = \sum_{k}^{N-1}{}_{1} Y_{\ell,k}[b_{j-1}Y_{j-1,k} + a_j Y_{j,k} + b_j Y_{j+1,k}] = b_j \delta_{\ell,j+1} .$$

Third, we will demonstrate that the data $\{\mu,y\}$ characterizes J. It will be sufficient to prove that the matrices J,Y constructed by this algorithm satisfy (1). Step 6 implies that $JY = YD$ if we can show that the numbers

$$Y_{N,j} \equiv (\mu_j - a_{N-1})Y_{N-1,j} - b_{N-2}Y_{N-2,j} \qquad \forall\, j$$

are zero. The techniques presented in the previous paragraph can be used to demonstrate that

$$\sum_{k}^{N-1}{}_{1} Y_{N,k} Y_{j,k} = 0 \qquad \text{for } j = 1,\cdots,N-1 .$$

Since the rows of Y form a real, orthonormal basis we infer that

$$Y_{N,j} = 0 \qquad \forall\, j$$

■

<u>Lemma 2.3</u>: Each set of compatible data characterizes at most one Jacobi matrix.

Proof: Let $\hat{J}$ be any Jacobi matrix characterized by the compatible data $\{\mu,y\}$. Then $y_1,\cdots,y_{N-1}$ are the first components of a set $\hat{Y}_1,\cdots,\hat{Y}_{N-1}$ of real, orthonormal eigenvectors of $\hat{J}$ corresponding to the eigenvalues $\mu_1,\cdots,\mu_{N-1}$. If $\hat{Y}$ denotes the matrix whose $j^{\underline{th}}$ column is $\hat{Y}_j$ then $\hat{Y}$ is an orthogonal matrix and

$$\hat{J}\,\hat{Y} = \hat{Y}\,D \qquad \text{where} \qquad D = \begin{bmatrix} \mu_1 & & 0 \\ & \ddots & \\ 0 & & \mu_{N-1} \end{bmatrix} .$$

We will now prove that the entries $(\hat{a},\hat{b})$ of $\hat{J}$ are identical to the entries (a,b) of the Jacobi matrix J computed by the algorithm presented in Theorem 2.

The entries $\hat{Y}_{i,j}$ of $\hat{Y}$ satisfy the orthonormality relations

$$\sum_{k}^{N-1}{}_{1} \hat{Y}_{i,k}\hat{Y}_{j,k} = \delta_{i,j} \qquad \forall\, i,j$$

because $\hat{Y}\hat{Y}^T = I$. The entries $\hat{Y}_{i,j}$ of Y also satisfy the recurrence relation

$$\hat{b}_{i-1}\hat{Y}_{i-1,j} + \hat{a}_i\hat{Y}_{i,j} + \hat{b}_i\hat{Y}_{i+1,j} = \mu_j\hat{Y}_{i,j} \qquad \forall\, i,j$$

where

$$\hat{Y}_{0,j} = 0 \quad \text{and} \quad \hat{Y}_{N,j} = 0 \qquad \forall\, j$$

because $\hat{J}\,\hat{Y} = \hat{Y}\,D$. When the recurrence relation is multiplied by $\hat{Y}_{i,j}$ and the result is summed over j we find, using the orthonormality relations, that

$$\hat{a}_i = \sum_{k}^{N-1}{}_{1} \mu_k \hat{Y}^2_{i,k} .$$

The recurrence relation also implies that

$$\hat{Y}_{i+1,j} = \frac{1}{\hat{b}_i}[(\mu_j-\hat{a}_i)\hat{Y}_{i,j} - \hat{b}_{i-1}\hat{Y}_{i-1,j}] \qquad \forall j$$

and, when this identity is squared and the result is summed over j , the orthonormality relations imply that

$$\hat{b}_i^2 = \sum_{k=1}^{N-1} [(\mu_k - \hat{a}_i)\hat{Y}_{i,k} - \hat{b}_{i-1}\hat{Y}_{i-1,k}]^2 .$$

Starting with the fact that

$$\hat{Y}_{1,j} = y_j \qquad \forall j$$

it is easily shown by induction, following the sequence of computations presented in the algorithm, that the entries $(\hat{a},\hat{b})$ of $\hat{J}$ are identical to the entries (a,b) of J . ■

3. Spectral Properties of Periodic Jacobi Matrices. In this section we will consider the spectral properties of the periodic Jacobi matrix

$$L = \begin{bmatrix} a_1 & b_1 & & & b_N \\ b_1 & \ddots & \ddots & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & \ddots & \ddots & b_{N-1} \\ b_N & & & b_{N-1} & a_N \end{bmatrix} \qquad \text{where} \quad b_i > 0 \quad \forall i .$$

Throughout this section we will use J to represent the Jacobi matrix obtained by deleting from L the last row and column.

Observe that L is a real, symmetric matrix. Consequently L has real eigenvalues and corresponding set of real, orthonormal eigenvectors [13, 14]. Let z be an eigenvector of L corresponding to the eigenvalue λ . Then the components z_i of z form a nontrivial solution of the recurrence relation $(b_0 \equiv b_N)$

$$b_{i-1}z_{i-1} + a_i z_i + b_i z_{i+1} = \lambda z_i \qquad \forall i$$

which satisfies the boundary conditions

$$z_N = z_0 \quad \text{and} \quad z_{N+1} = z_1 .$$

By analogy with Floquet theory, which analyzes the analogous problem for ordinary differential equations [11], let us consider the nontrivial solutions of the recurrence relation which satisfy the boundary conditions

$$z_N = \rho z_0 \quad \text{and} \quad z_{N+1} = \rho z_1 .$$

Here the parameter ρ is called the Floquet multiplier of z . This problem has only the trivial solution when $\rho = 0$, while for $\rho \neq 0$ a nontrivial solution exists if and only if λ is an eigenvalue of the matrix

$$L_\rho = \begin{bmatrix} a_1 & b_1 & & & \frac{1}{\rho}b_N \\ b_1 & \ddots & \ddots & 0 & \\ & \ddots & \ddots & \ddots & \\ & 0 & \ddots & \ddots & b_{N-1} \\ \rho b_N & & & b_{N-1} & a_N \end{bmatrix} .$$

With these facts in mind let us introduce the following:

Definition 3: Let J be characterized by the data $\{\mu, y\}$ and have $\omega(\mu)$ as its characteristic polynomial. Then the Floquet multipliers $\rho_1, \cdots, \rho_{N-1}$ of L corresponding to $\mu_1, \cdots, \mu_{N-1}$ are the numbers defined by the relation

$$b_1 \cdots b_N = -\rho_j \omega'(\mu_j)\, b_N^2 y_j^2 \qquad \forall\, j . \tag{6}$$

Theorem 4: The characteristic polynomial of L_ρ admits the representation

$$\det(\lambda I - L_\rho) = b_1 \cdots b_N \,\{\Delta(\lambda) - (\rho + \frac{1}{\rho})\} \tag{7}$$

where $\Delta(\lambda)$, called the discriminant of L, is independent of ρ. The Floquet multipliers $\rho_1, \cdots, \rho_{N-1}$ of L corresponding to the eigenvalues $\mu_1, \cdots, \mu_{N-1}$ of J satisfy the relation

$$\Delta(\mu_j) = \rho_j + \frac{1}{\rho_j} \qquad \forall\, j . \tag{8}$$

The eigenvalues $\lambda_1, \cdots, \lambda_N$ of L are real and can be ordered so that

$$\lambda_1 > \lambda_2 \geq \lambda_3 > \lambda_4 \geq \lambda_5 > \cdots .$$

Proof: Using elementary properties of determinants it is not hard to demonstrate that

$$\frac{d}{d_\rho} \det(\lambda I - L_\rho) = -b_1 \cdots b_N \,(1 - \frac{1}{\rho^2}) .$$

When both sides are integrated with respect to ρ we find that

$$\det\,(\lambda I - L_\rho) = b_1 \cdots b_N \{\Delta(\lambda) - (\rho + \frac{1}{\rho})\} .$$

Of course the constant of integration $b_1 \cdots b_N\, \Delta(\lambda)$ is necessarily independent of ρ.

Let J be characterized by the data $\{\mu, y\}$. Then $y_1, \cdots, y_{N-1}$ are the first components of a set $Y_1, \cdots, Y_{N-1}$ of real, orthonormal eigenvectors of J corresponding to its eigenvalues $\mu_1, \cdots, \mu_{N-1}$. Let $Y_{i,j}$ denote the ith component of Y_j. From the definition (6) of the Floquet multipliers and the identity (4) we infer that

$$\rho_j = -\frac{b_{N-1} Y_{N-1,j}}{b_N Y_{1,j}} \qquad \forall\, j$$

and so

$$L_{\rho_j}\begin{bmatrix} Y_j \\ \\ \\ 0 \end{bmatrix} = \mu_j \begin{bmatrix} Y_j \\ \\ \\ 0 \end{bmatrix} \qquad \forall j .$$

Consequently μ_j is an eigenvalue of L_{ρ_j} for each j and we infer from (7) that (8) is true.

From the definition (6) of the Floquet multipliers we deduce that

$$\omega'(\mu_j)\rho_j < 0 \qquad \forall j .$$

When the eigenvalues of J are ordered so that

$$\mu_1 > \cdots > \mu_{N-1}$$

then we infer from (8) that

$$(-1)^j \Delta(\mu_j) \geq 2 \qquad \forall j$$

because the magnitude of $\rho + \frac{1}{\rho}$ is never less than two. Consequently the eigenvalues $\lambda_1,\cdots,\lambda_N$ of L, which are the roots of $\Delta(\lambda) = 2$, are real and can be ordered so that

$$\lambda_1 > \lambda_2 \geq \lambda_3 > \lambda_4 \geq \lambda_5 > \cdots$$

because the coefficient $(b_1 \cdots b_N)^{-1}$ of λ^N in $\Delta(\lambda)$ is positive. ■

A typical discriminant for a periodic Jocobi matrix L of order $N = 6$ is illustrated in Figure 1. In this figure we depict the relationship between the eigenvalues $\lambda_1,\cdots,\lambda_N$ of L and the Floquet multipliers $\rho_1,\cdots,\rho_{N-1}$ of L corresponding to the eigenvalues $\mu_1,\cdots,\mu_{N-1}$ of J .

Let us introduce the following:

<u>Definition 5</u>: (a) The periodic Jacobi matrix L is <u>characterized</u> by the data $\{A,B,\mu,\rho\}$ if and only if

(1) $A = a_1 + \cdots + a_N$,

(2) $B = b_1 \cdots b_N$,

(3) $\mu_1,\cdots,\mu_{N-1}$ are the eigenvalues of J , and

(4) $\rho_1,\cdots,\rho_{N-1}$ are the Floquet multipliers of L corresponding to $\mu_1,\cdots,\mu_{N-1}$.

(b) The data $\{A,B,\mu,\rho\}$ is <u>compatible</u> if and only if

(1) A is a real number,

(2) B is a real, positive number,

(3) $\mu_1,\cdots,\mu_{N-1}$ are real, distinct numbers, and

(4) $\rho_1,\cdots,\rho_{N-1}$ are real numbers which satisfy $\omega'(\mu_j)\rho_j < 0 \quad \forall j$ with $\omega(\mu) = (\mu-\mu_1)\cdots(\mu-\mu_{N-1})$.

We feel justified in using the words "characterize" and "compatible" in this manner because the following theorem is true.

Theorem 6: Data characterizing a periodic Jacobi matrix is compatible. Furthermore, each set of compatible data $\{A,B,\mu,\rho\}$ characterizes a unique periodic Jacobi matrix L. The entries (a,b) of this periodic Jacobi matrix are computed by the algorithm:

1. $$b_N^2 = -\sum_{k=1}^{N-1} \frac{B}{\rho_k \omega'(\mu_k)}$$

2. $$y_j = \frac{1}{b_N}\sqrt{-\frac{B}{\rho_j \omega'(\mu_j)}} \qquad \forall\ j$$

3. Recover J from the data $\{\mu,y\}$

4. $$b_{N-1} = \frac{B}{b_1 \cdots b_{N-2} b_N}$$

5. $$a_N = A - (a_1 + \cdots + a_{N-1})$$

with $\omega(\mu) = (\mu-\mu_1)\cdots(\mu-\mu_{N-1})$.

Proof: The proof of this theorem will be presented as a sequence of three lemmas.

■

Lemma 6.1: Data characterizing a periodic Jacobi matrix is compatible.

Proof: Let the periodic Jacobi matrix L be characterized by the data $\{A,B,\mu,\rho\}$. Clearly A is a real number because it is a sum of real numbers, while B is a real, positive number because it is a product of real, positive numbers. The μ's are real, distinct numbers because they are the eigenvalues of the Jacobi matrix J. while the definition (5) of the ρ's makes it obvious that they are real, nonzero numbers which satisfy $\omega'(\mu_j)\rho_j < 0$ for all j because $\omega(\mu)$ is also the characteristic polynomial of J.

■

Lemma 6.2: Given compatible data $\{A,B,\mu,\rho\}$ the algorithm of Theorem 6 computes the entries (a,b) of a periodic Jacobi matrix L characterized by the data $\{A,B,\mu,\rho\}$.

Proof: The data $\{\mu,y\}$ used in step 3 is compatible, therefore it is clear that this algorithm computes the entries (a,b) of some periodic Jacobi matrix L. Let L be characterized by the data $\{\hat{A},\hat{B},\hat{\mu},\hat{\rho}\}$. From steps 4 and 5 it is clear that $\hat{A} = A$ and $\hat{B} = B$. In view of Theorem 2 we know that J is characterized by the data $\{\mu,y\}$. Therefore $\hat{\mu}_j = \mu_j$ for all j and from the definition of the Floquet multipliers we know that

$$B = -\hat{\rho}_j \omega'(\mu_j)\ b_N^2\, y_j^2 \qquad \forall\ j\ .$$

Step 2 therefore implies that $\hat{\rho}_j = \rho_j$ for all j.

■

Lemma 6.3: Each set of compatible data characterizes at most one periodic Jacobi matrix.

Proof: Let $\hat{L}$ be any periodic Jacobi matrix characterized by the compatible data $\{A,B,\mu,\rho\}$. Let the Jacobi matrix $\hat{J}$, obtained by deleting from $\hat{L}$ the last row and column, be characterized by the data $\{\mu,\hat{y}\}$. Without loss of generality we may assume that each $\hat{y}_j$ is positive, for if $\hat{y}_j$ is the first component of an eigenvector $\hat{Y}_j$ of $\hat{J}$ then $-\hat{y}_j$ is the first component of the eigenvector $-\hat{Y}_j$ of $\hat{J}$. We will now prove that the entries $(\hat{a},\hat{b})$ of $\hat{L}$ are identical to the entries (a,b) of the periodic Jacobi matrix L constructed by the algorithm of Theorem 6.

By definition the Floquet multipliers $\hat{\rho}_1,\cdots,\hat{\rho}_{N-1}$ of $\hat{L}$ corresponding to $\mu_1,\cdots,\mu_{N-1}$ satisfy the relationship

$$B = -\rho_j \omega'(\mu_j) \ \hat{b}_N^2 \ \hat{y}_j^2 \qquad \forall\ j .$$

The sum of the squares of the $\hat{y}$'s equals one because the data $\{\mu,\hat{y}\}$ is compatible, therefore

$$\hat{b}_N^2 = -\sum_{k=1}^{N-1} \frac{B}{\rho_k \omega'(\mu_k)} \quad \text{and} \quad \hat{y}_j = \frac{1}{b_N}\sqrt{-\frac{B}{\rho_j \omega'(\mu_j)}} \quad \forall\ j .$$

In view of steps 1 and 2 we infer that $\hat{b}_N = b_N$ and $\hat{y}_j = y_j$ for all j. Since both $\hat{J}$ and J are characterized by the same data then Theorem 2 implies that $\hat{J} = J$. Finally, steps 4 and 5 imply that $\hat{b}_{N-1} = b_{N-1}$ and $\hat{a}_N = a_N$.

■

4. Periodic Jacobi Matrices with Prescribed Spectra.

With these basic facts established let us now consider how we can characterize the family of periodic Jacobi matrices whose eigenvalues are $\lambda_1,\cdots,\lambda_N$.

Let L be a periodic Jacobi matrix characterized by the data $\{A,B,\mu,\rho\}$. Then $\lambda_1,\cdots,\lambda_N$ are the eigenvalues of L if and only if the discriminant $\Delta(\lambda)$ of L admits the representation

$$\Delta(\lambda) = 2 + \frac{1}{B}(\lambda-\lambda_1)\cdots(\lambda-\lambda_N) .$$

Therefore the problem of characterizing the family of periodic Jacobi matrices with prescribed spectra is intimately related to the problem of characterizing the family of periodic Jacobi matrices with prescribed discriminant. Let us introduce the following:

Definition 7: For each polynomial $p(\lambda)$ let $\mathcal{F}(p)$ denote the family of periodic Jacobi matrices whose discriminant is $p(\lambda)$.

The problem of characterizing which periodic Jacobi matrices belong to $\mathcal{F}(p)$ is answered in the following:

Theorem 8: Let $p(\lambda)$ be a polynomial of degree N . The data $\{A,B,\mu,\rho\}$ characterizes a member of $\mathcal{F}(p)$ if and only if:

(1) the data $\{A,B,\mu,\rho\}$ is compatible,

(2) $p(\lambda) = \frac{1}{B}\,[\lambda^N - A\,\lambda^{N-1} + \text{lower powers of } \lambda\,]$, and

(3) $p(\mu_j) = \rho_j + \frac{1}{\rho_j} \quad \forall\, j$.

Furthermore, $\mathcal{F}(p)$ is nonempty if and only if

(4) the coefficient of λ^N in $p(\lambda)$ is positive, and

(5) $p(\lambda)$ has local extrema at N-1 real, distinct points $\nu_1 > \cdots > \nu_{N-1}$ with $(-1)^j\, p(\nu_j) \geq 2 \quad \forall\, j$.

Proof: The proof of this theorem will be presented as a sequence of two lemmas.

■

Lemma 8.1: The data $\{A,B,\mu,\rho\}$ characterizes a member of $\mathcal{F}(p)$ if and only if conditions (1,2,3) of Theorem 8 are satisfied.

Proof: If the data $\{A,B,\mu,\rho\}$ characterizes a member of $\mathcal{F}(p)$ then Theorems 4 and 6 demonstrate that conditions (1,2,3) of Theorem 8 are satisfied.

Let us now suppose that conditions (1,2,3) of Theorem 8 are satisfied. Let $\Delta(\lambda)$ be the discriminant of the periodic Jacobi matrix characterized by the data $\{A,B,\mu,\rho\}$. Now

$$q(\lambda) \equiv \Delta(\lambda) - p(\lambda)$$

is a polynomial of degree N-2 because the coefficients of λ^{N-1}, λ^N in $\Delta(\lambda)$, $p(\lambda)$ agree. Theorem 4 also implies that $q(\mu_j) = 0 \quad \forall\, j$ and so

$$q(\lambda) \equiv 0$$

because the only polynomial of degree N-2 which is zero at N-1 distinct points is the trivial polynomial. Consequently the data $\{A,B,\mu,\rho\}$ characterizes a member of $\mathcal{F}(p)$.

■

Lemma 8.2: $\mathcal{F}(p)$ is nonempty if and only if conditions (4,5) of Theorem 8 are satisfied.

Proof: If $\mathcal{F}(p)$ is nonempty then Lemma 8.1 and the mean-value theorem can be used to demonstrate that conditions (4,5) of Theorem 8 are satisfied.

Let us now suppose that conditions (4,5) of Theorem 8 are satisfied. Let A,B be determined so that

$$p(\lambda) = \frac{1}{B}[\lambda^N - A\,\lambda^{N-1} + \text{lower powers of } \lambda\,]$$

and $\rho_1, \cdots, \rho_{N-1}$ be solutions of

$$p(\nu_j) = \rho_j + \frac{1}{\rho_j} \qquad \forall\, j\ .$$

Then the data $\{A,B,\nu,\rho\}$ is compatible and from Lemma 8.1 we infer that the data $\{A,B,\nu,\rho\}$ characterizes a member of $\mathcal{F}(p)$.

■

Using Theorem 8 it is not hard to prove the following:

Corollary 9: The periodic Jacobi matrix L has $\lambda_1, \cdots, \lambda_N$ as its eigenvalues if and only if

$$L \in \bigcup_{B>0} \mathcal{F}(\Delta_B)$$

where $\Delta_B(\lambda) \equiv 2 + \frac{1}{B}(\lambda-\lambda_1)\cdots(\lambda-\lambda_N)$. Furthermore, there is a periodic Jacobi matrix with $\lambda_1, \cdots, \lambda_N$ as its eigenvalues if and only if the numbers $\lambda_1, \cdots, \lambda_N$ can be ordered so that

$$\lambda_1 > \lambda_2 \geq \lambda_3 > \lambda_4 \geq \lambda_5 > \cdots \qquad .$$

5. Numerical Experiments. Let us now present the results of several numerical experiments. These experiments were carried out on a UNIVAC 1110 in single precision floating point arithmetic (27 bit mantissa) using FORTRAN versions of the algorithms presented in Theorems 2 and 6 .

In the first experiment we test the algorithm presented in Theorem 2. The results of this experiment are presented in Table 1. Observe that this algorithm has difficulty in recovering the Jacobi matrix described in Example 3.

Experiment 1:

1. Select a Jacobi matrix J of order $N-1$.
2. Compute the data $\{\mu,y\}$ characterizing J [13, 14, 15]:
   (a) use bisection to compute the μ's , and
   (b) use inverse iteration to compute the y's .
3. Use the algorithm presented in Theorem 2 to reconstruct the Jacobi matrix $\hat{J}$ characterized by the data $\{\mu,y\}$.
4. Output the error $\|J - \hat{J}\|$ where

$$\|A\| = \max_{i,j} |a_{i,j}|\ .$$

In the second experiment we test the algorithm presented in Theorem 6. The results of this experiment are presented in Table 2. Observe that the Jacobi matrices used in the examples of Experiment 1 are obtained by deleting the last row and column from the periodic Jacobi matrix used in the corresponding examples of Experiment 2.

Experiment 2:

1. Select a periodic Jacobi matrix L of order N.
2. Compute the data $\{A,B,\mu,\rho\}$ characterizing L:
   (a) use the obvious sum to compute A,
   (b) use the obvious product to compute B,
   (c) compute the data $\{\mu,y\}$ characterizing J as described in Step 2 of Experiment 1, and
   (d) compute the ρ's using Equation (6).
3. Use the algorithm presented in Theorem 6 to reconstruct the periodic Jacobi matrix $\hat{L}$ characterized by the data $\{A,B,\mu,\rho\}$.
4. Output the error $\|L-\hat{L}\|$ where

$$\|A\| = \max_{i,j} |a_{ij}| .$$

In both of these experiments we have not worked with matrices of order $N > 30$. The reason why we have not worked with matrices of order $N > 30$ may be explained as follows. In Example 2 of Experiment 2 some of the components of y in the data $\{\mu,y\}$ become smaller as N increases. For example, the smallest component of y changes from 2×10^{-9} for $N = 15$ to 2×10^{-20} for $N = 30$. Since the Floquet multipliers ρ depend on the squares of the data y we will run into underflow problems when $N > 30$. The immediate remedy for this underflow problem is the use of logarithms in the computation of the Floquet multipliers. However, underflow also occurs in the computation of y when $N > 55$, consequently the use of logarithms is not a panacea.

6. Comments. Let $\tilde{\omega}(\mu)$ be the characteristic polynomial of the Jacobi matrix $\tilde{J}$ obtained from J by deleting the first row and column. By comparing the entries of (2) in row 1, column 1 we find that

$$\tilde{\omega}(\mu) = \sum_{k}^{N-1} {}_{1} \frac{\omega(\mu)}{\mu - \mu_k} Y_{1,k}^{2} .$$

This identity was used by Stieltjes in his study of inverse eigenvalue problems. As μ tends to μ_j we deduce that

$$\tilde{\omega}(\mu_j) = \omega'(\mu_j)\, Y_{1,j}^{2} \qquad \forall j .$$

From this identity we infer that the eigenvalues of $\tilde{J}$ strictly interlace those of J. Furthermore, from the eigenvalues of J and $\tilde{J}$ we can recover the data $\{\mu,y\}$ characterizing J and hence J itself.

It is interesting to note that the algorithm presented in Theorem 2 is used in some versions of the implicit shift QR algorithm [13]. These versions of the QR algorithm make use of the fact that if $B = QAQ^H$, where B is an unreduced upper Hessenberg matrix and Q is a unitary matrix, then the entries of B and Q are uniquely determined from the entries of A and the entries in the first row of Q. In our application $A = D$, $B = J$ and $Q = Y$.

We can also recover the Jacobi matrix J from the eigenvalues and the last components of the corresponding real, orthonormal eigenvectors of J. To understand why let us consider the permutation matrix

$$S = \begin{bmatrix} 0 & & 1 \\ & \iddots & \\ 1 & & 0 \end{bmatrix} .$$

We find that $S^2 = I$, therefore from Equation (1) we deduce that

$$(SJS)(SY) = (SY)D .$$

Consequently the algorithm presented in Theorem 2 states that the entries of

$$SJS = \begin{bmatrix} a_{N-1} & b_{N-2} & & 0 \\ b_{N-2} & \ddots & \ddots & \\ & \ddots & \ddots & b_1 \\ 0 & & b_1 & a_1 \end{bmatrix}$$

can be recovered from the entries of D and the entries in the first row of SY, that is the last row of Y.

It also appears that Theorem 2 can be extended to some class of band matrices. For example, let the real, symmetric matrix

$$K = \begin{bmatrix} a_1 & b_1 & c_1 & & 0 \\ b_1 & \ddots & \ddots & \ddots & \\ c_1 & \ddots & \ddots & \ddots & c_{N-3} \\ & \ddots & \ddots & \ddots & b_{N-2} \\ 0 & & c_{N-3} & b_{N-2} & a_{N-1} \end{bmatrix} \quad \text{where } c_i > 0 \ \forall\, i .$$

have $\mu_1, \cdots, \mu_{N-1}$ as eigenvalues and $Y_1, \cdots, Y_{N-1}$ as the corresponding set of real, orthonormal eigenvectors. If Y denotes the matrix whose $j\underline{th}$ column is Y_j then Y is an orthogonal matrix and

$$KY = YD \quad \text{where} \quad D = \begin{bmatrix} \mu_1 & & 0 \\ & \ddots & \\ 0 & & \mu_{N-1} \end{bmatrix} .$$

Following the argument presented in Lemma 2.3 we arrive at an algorithm which recovers K from the entries in D and in the first two rows of Y .

The paper by Golub and Welsch [7] outlines how one can modify the usual QR algorithm and compute directly the data $\{\mu, y\}$ characterizing a Jacobi matrix J. Furthermore, their paper also presents a matrix version of the celebrated Gelfand-Levitan solution to the inverse eigenvalue problem for a class of Sturm-Liouville problems.

The paper by Kammerer [10] describes an algorithm that can be used to construct a discriminant whose "shape" is prescribed. By the "shape" of a discriminant we are referring to the value of the discriminant at each of its N-1 real, distinct local extrema. For applications of Kammerer's algorithm to the periodic Toda lattice we refer the reader to the forthcoming paper [4].

Useful information concerning properties of periodic Jacobi matrices is contained in [1]. We would also like to state that the analysis presented in Section 3 can be extended in the same generality to "anti-periodic" Jacobi matrices of the form

$$L_{-1} = \begin{bmatrix} a_1 & b_1 & & -b_N \\ b_1 & \ddots & \ddots \quad 0 & \\ & 0 \quad \ddots & \ddots & b_{N-1} \\ -b_N & & b_{N-1} & a_N \end{bmatrix} \qquad \text{where } b_i > 0 \quad \forall\, i .$$

7. Acknowledgements. The author would like to thank Professor C. de Boor, H. Flaschka, G. Golub and D. McLaughlin for several informative discussions. Indeed, the results presented in this paper arose from work done with Flaschka and McLaughlin [4] on the periodic Toda lattice while the algorithm presented in Theorem 2 is essentially the same algorithm presented by de Boor and Golub [3]. The derivation of the spectral properties of periodic Jacobi matrices depends quite heavily on the discrete version of Floquet theory as presented by Flaschka and McLaughlin [5,6] and by van Moerbeke [12].

The author would also like to thank the agencies which support the Mathematics Research Center at the University of Wisconsin.

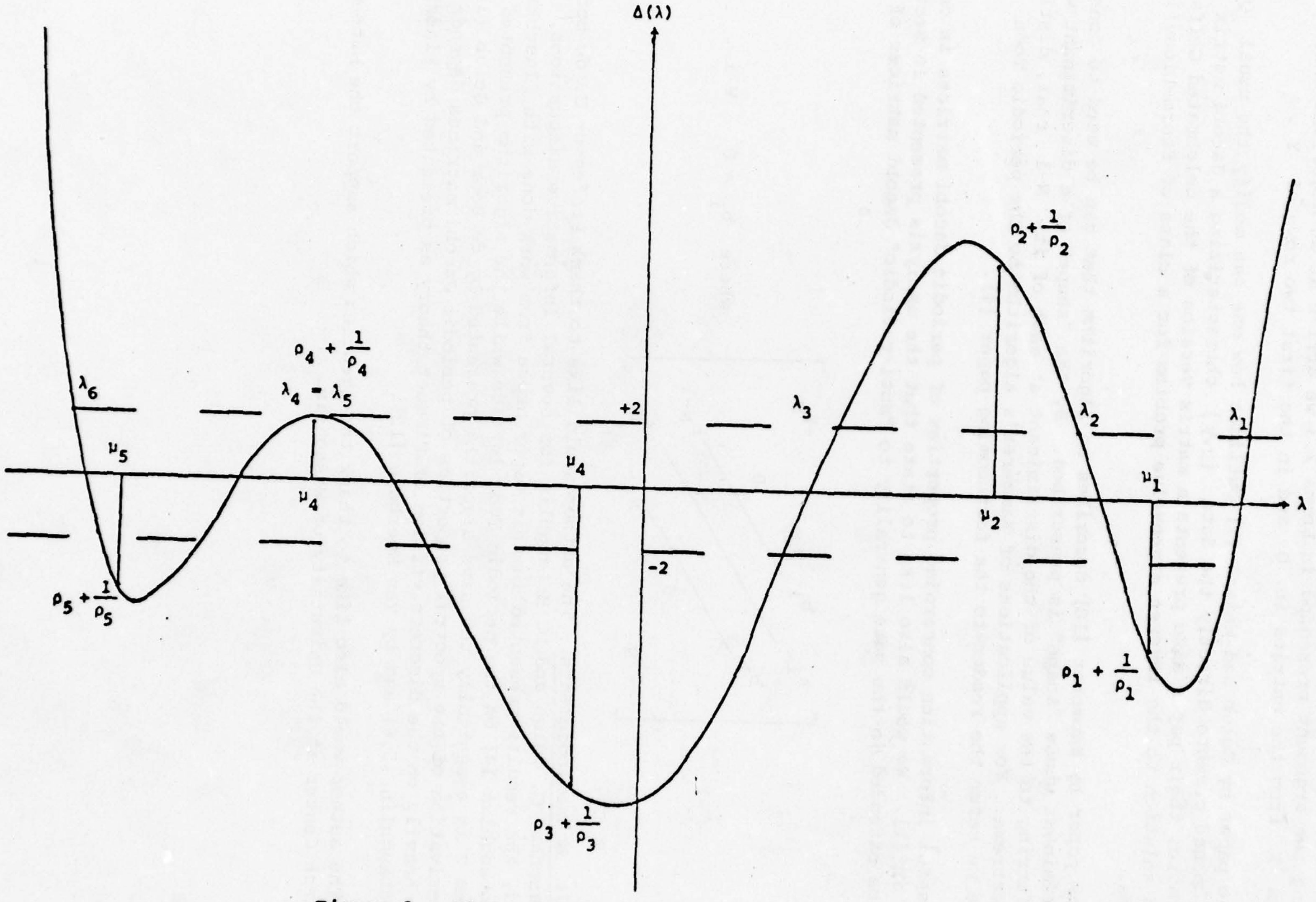


Figure 1: Plot of a Typical Discriminant, N=6

Example 1:

A(I) = -2 I = 1,···,N-1

B(I) = 1 I = 1,···,N-2

N	Error
5	4×10^{-8}
10	2×10^{-7}
15	5×10^{-7}
20	2×10^{-7}
25	2×10^{-7}
30	6×10^{-7}

Example 2:

A(I) = (N+1-I)/N-2 I = 1,···,N-1

B(I) = 1-(N-I)/N I = 1,···,N-2

N	Error
5	4×10^{-8}
10	1×10^{-7}
15	4×10^{-7}
20	3×10^{-7}
25	3×10^{-7}
30	9×10^{-7}

Table 1: Results of Experiment 1.

Example 3:

$$A(I) = I/N - 2 \qquad I = 1,\cdots,N-1$$

$$B(I) = 1 - I/N \qquad I = 1,\cdots,N-2$$

N	Error
5	1×10^{-7}
10	3×10^{-7}
15	2×10^{-4}
20	2×10^{0}
25	2×10^{0}
30	1×10^{0}

Table 1: Results of Experiment 1.

Example 1:

$A(I) = -2 \qquad I = 1,\cdots,N-1$

$B(I) = 1 \qquad I = 1,\cdots,N-2$

$A(N) = 0$

$B(N-1) = B(N) = 1$

N	Error
5	9×10^{-8}
10	5×10^{-7}
15	1×10^{-6}
20	2×10^{-6}
25	3×10^{-6}
30	5×10^{-6}

Example 2:

$A(I) = (N+1-I)/N-2 \qquad I = 1,\cdots,N-1$

$B(I) = 1-(N-I)/N \qquad i = 1,\cdots,N-2$

$A(N) = 0$

$B(N-1) = B(N) = 1$

N	Error
5	1×10^{-7}
10	2×10^{-7}
15	4×10^{-7}
20	3×10^{-7}
25	6×10^{-7}
30	4×10^{-7}

Table 2 - Results of Experiment 2

Example 3:

$A(I) = I/N-2 \qquad I = 1,\cdots,N-1$

$B(I) = 1 - I/N \qquad I = 1,\cdots,N-2$

$A(N) = 0$

$B(n-1) = B(N) = 1$

N	Error
5	4×10^{-8}
10	1×10^{-7}
15	1×10^{-3}
20	2×10^{0}
25	4×10^{0}
30	5×10^{0}

Table 2. Results of Experiment 2.

References

1. A. Bjorck and G. Golub, Eigenproblems for Matrices Associated with Periodic Boundary Conditions, SIAM Review 19 (1977), 5-16.

2. D. Boley and G. Golub, The Matrix Inverse Eigenvalue Problem for Periodic Jacobi Matrices, Computer Sciences Department, Stanford, California, STAN-CS-78-684 (1978).

3. C. de Boor and G. Golub, The Numerically Stable Reconstruction of a Jacobi Matrix from Spectral Data, MRC-TSR-1727, Mathematics Research Center, University of Wisconsin, Madison, Wisconsin 53706.

4. W. Ferguson, H. Flaschka, and D. W. McLaughlin, Nonlinear Normal Modes for the Periodic Toda Lattice, in preparation.

5. H. Flaschka, The Toda Lattice, Phys. Rev. B 9 (1974), 1924-1925.

6. H. Flaschka and D. W. McLaughlin, Cannonically Conjugate Variables for the Korteweg-deVries Equation and the Toda Lattice with Periodic Boundary Conditions, Prog. Theor. Phys. 55 (1976), 438-456.

7. G. Golub and J. Welsch, Calculation of Gauss Quadrature Rules, Math. Comp. 23 (1969), 221-230.

8. O. Hald, Inverse Eigenvalue Problems for Jacobi Matrices, Linear Algebra Appl. 14 (1976), 63-85.

9. H. Hochstadt, On the Construction of a Jacobi Matrix from Spectral Data, Linear Algebra Appl. 8 (1974), 435-446.

10. W. Kammerer, Polynomial Approximations to Finitely Oscillating Functions, Math. Comp. 15 (1961), 115-119.

11. W. Magnes and S. Winkler, Hill's Equation, Wiley-Interscience, New York (1965).

12. P. van Moerbeke, The Spectrum of Jacobi Matrices, Invent. Math. 37 (1976), 45-81.

13. G. Stewart, Introduction to Matrix Computations, Academic Press, New York (1973).

14. J. H. Wilkinson, The Algebraic Eigenvalue Problem, Oxford University Press, London (1965).

15. J. H. Wilkinson and C. Reinsch, Linear Algebra, Springer-Verlag, New York (1971).

APPROXIMATING MULTIVARIATE FUNCTIONS BY COMBINATIONS OF UNIVARIATE ONES

E. W. Cheney
Department of Mathematics
University of Texas
Austin, Texas 78712

ABSTRACT. This is a brief introduction to some aspects of the topic in the title, emphasizing mainly the approximation problem

(1) $$f(x,y) \simeq g(x) + h(y)$$

1. INTRODUCTION. Many of the results in this branch of approximation theory have been or can be established for functions of any number of variables. For simplicity, the discussion here is limited to bivariate functions. The model problem, from which many interesting other problems are derived, is to represent exactly or approximately a given function of two variables by means of simpler functions built-up by combining univariate functions. For example, an empirical function $f(x,y)$ may be known only in the form of a table containing thousands of entries. If it is necessary to program a computer subroutine for f, it would be very advantageous to know that f had a representation

(2) $$f(x,y) = \phi\,[g(x) + h(y)]$$

If this were true, then the subroutine problem would be much simpler. In this case the apparent complexity of f, as evidenced by the large table of its values, was illusory, for the equation (2) shows f to be rather simple, and capable of being tabulated by means of three univariate function tables.

Functions of the type appearing on the right side of equation (2) are termed nomographic. They can be represented graphically by a simple nomogram or alignment chart. For an example of a nomogram, see Fig.1. This represents the function $f(x,y) = \exp(\sin x + y^2)$. In order to use it, locate x on the vertical x-axis and y on the y-axis. A straight line through these points intersects the f-axis at $f(x,y)$.

(*) Author supported by the Army Research Office.

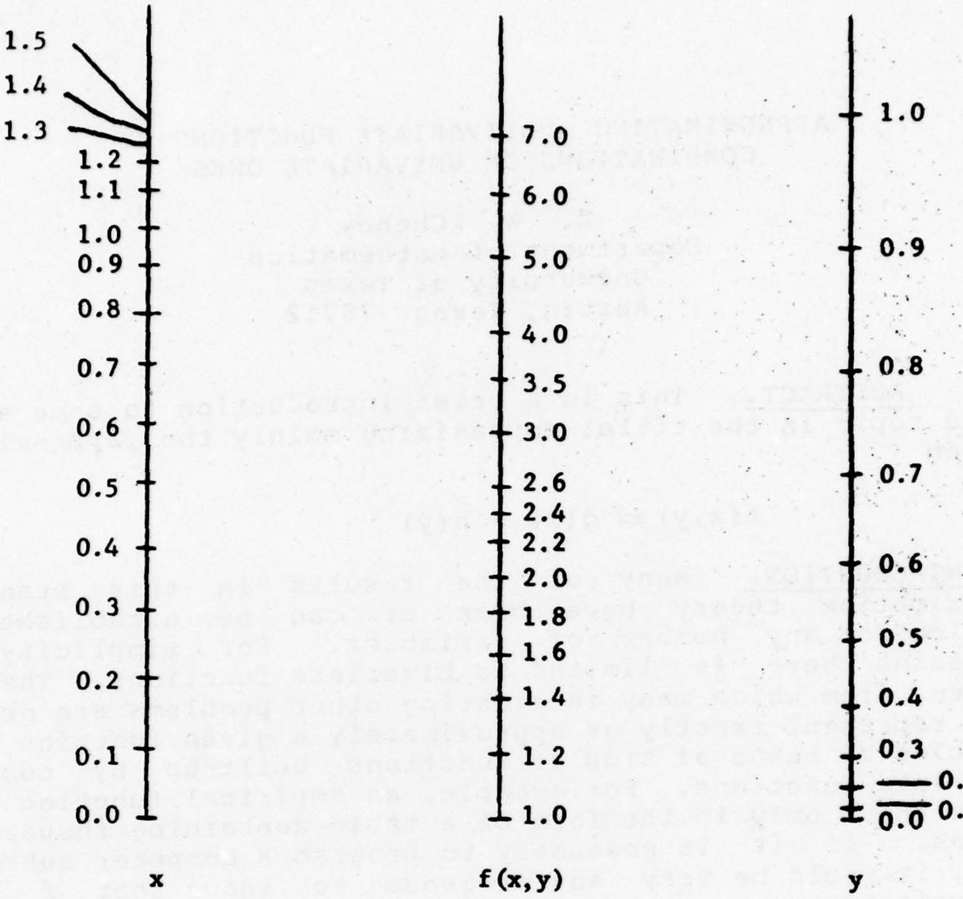


Figure 1. Nomogram for $\exp(\sin x + y^2)$

$0 \leq x \leq 1.5$ and $0 \leq y \leq 1.0$

It turns out that nomographic functions are the building- blocks for all multivariate functions. The following theorem asserting this is due to Kolmogoroff, 1955.

THEOREM. Every continuous function of two variables defined on a compact set in the plane is the sum of at most 5 continuous nomographic functions.

Many refinements have been made in Kolmogoroff's Theorem, and the interested reader should consult [17]. Our purpose in mentioning Kolmogoroff's theorem here is to focus attention on the nomographic functions and their power as approximating functions. This point of view has received much attention by R.C.Buck in a series of papers [5,6,7,8].

A more basic type of approximation consists in selecting only two univariate functions to approximate f in the form given in Equation (1), and we restrict our attention to that problem in the remainder of this note.

2.THE L-2 APPROXIMATION PROBLEM. We begin with that problem because it is elementary. Suppose that f is a function belonging to class L-2 on the unit square. We seek L-2 functions g(x) and h(y) to minimize the expression

$$\|f-g-h\|_2^2 = \int_0^1\int_0^1 [f(x,y)-g(x)-h(y)]^2\,dx\,dy$$

The solution is most readily obtained by defining two operators P and Q as follows:

$$(Pf)(x,y) = \int_0^1 f(x,t)\,dt$$

$$(Qf)(x,y) = \int_0^1 f(s,y)\,ds$$

Note that Pf is really a function of x alone, and Qf is really a function of y alone. It is easy to see that Pf is the best L-2 approximation of f by an L-2 function of x. A similar statement holds for Qf. Now define the "Boolean Sum" of P and Q:

$$B = P + Q(I-P)$$

In the present circumstances, P, Q, and B are all linear. Hence B = P + Q - QP. The operator B produces best approximations of the form g(x)+h(y), and so solves our problem. Thus the best approximation of f by the sum of L-2 functions of x and y separately is

$$(Bf)(x,y) = \int_0^1 f(x,t)\,dt + \int_0^1 [f(s,y) - \int_0^1 f(s,t)\,dt]\,ds$$

The foregoing remarks are actually just an illustration of the abstract theory of linear projection operators. The following general theorem applies to this situation.

THEOREM. If P and Q are bounded linear projections defined on a Banach space, and if P commutes with Q (i.e., PQ=QP), then the Boolean sum B=P+Q(I-P) is a projection onto the vector sum of the ranges of P and Q. In Hilbert space, B is orthogonal if P and Q are.

See [12] for further details and for related results in the case that P and Q do not commute.

3. THE UNIFORM APPROXIMATION PROBLEM. Here we assume that f is a function on a Cartesian product, X × Y, and we want to approximate it by g+h, where g is a function on X and h is a function on Y. In the uniform problem, we seek g and h so that the supremum norm will be a minimum:

$$\|f-g-h\|_\infty = \sup\left|f(x,y)-g(x)-h(y)\right|$$

Before describing the present status of this problem, we shall mention an area of numerical linear algebra in which it finds an application. Consider an m × n matrix A of positive numbers. The process of dividing each row and each column by certain positive constants is called "scaling". The resulting matrix is said to be optimally scaled if the ratio of its largest to its smallest element is as small as possible. Let the row divisors and column divisors be r_i and c_j respectively. The new matrix then has elements $a_{ij}r_i^{-1}c_j^{-1}$. For optimal scaling, the quantity

$$\max a_{ij}r_i^{-1}c_j^{-1} / \min a_{ij}\, r_i^{-1}c_j^{-1}$$

should be a minimum. By homogeniety, it is equivalent to minimize k subject to the inequalities

$$k \geqslant a_{ij}\, r_i^{-1}c_j^{-1} \geqslant 1$$

Using capital letters to denote the logarithms of these quantities, we have

$$K \geqslant A_{ij} - R_i - C_j \geqslant 0$$

and here it is desired to select R_i and C_j so that K will be a minimum. This is the same as minimizing the expression

$$\varepsilon = \|A-R-C\| = \max\left|A_{ij} - R_i - C_j\right|$$

and then observing that

$$2\varepsilon \geqslant A_{ij} - R_i - C_j + \varepsilon \geqslant 0$$

For further information on this application, see the papers [2,3,4, 10].

Returning now to the general problem, we attempt to proceed by analogy with the L-2 case. Let X and Y be compact Hausdorff spaces, such as closed and bounded intervals on the real line. Suppose that the functions involved are all continuous. The analogue of the orthogonal projections P and Q in Section 2 are now the proximity maps or metric projections from C(X Y) onto C(X) and C(Y). These are given by the formulas

$$(Pf)(x,y) = \frac{1}{2}\max_{t} f(x,t) + \frac{1}{2}\min_{t} f(x,t)$$

$$(Qf)(x,y) = \frac{1}{2}\max_{s} f(s,y) + \frac{1}{2}\min_{s} f(s,y)$$

as before, Pf is really a function of x and Qf a function of y. Their Boolean sum is B=P+Q(I-P). Since P and Q are not linear, this is not the same as P+Q-QP.

In this situation, Bf is not the best approximation of f from C(X)+C(Y). However, it is possible to use the method of iteration to obtain a best approximation. This is the fundamental discovery made by Diliberto and Straus in their seminal paper [9]. Explicitly, they defined a sequence by the recursive scheme

$$f_0 = f \qquad f_{n+1} = (I-B)f_n$$

More elegantly, $f_n = (I-B)^n f$. Then, combining results of [9] and [1], we have

THEOREM. If X and Y are compact Hausdorff spaces and if $f \in C(X\ Y)$, then the sequence $f-f_n$ converges uniformly to the sum of a continuous function of x and a continuous function of y which is a best approximation to f in the supremum norm. The convergence of $\|f_n\|$ is monotonically downward.

Other proofs of this theorem, which is rather deep, have been given in [14] and [15]. In [11] one finds estimates of the rapidity of convergence of f-f . It has been conjectured but never proved that for each f there is a $\lambda \in (0,1)$ and a number C such that

$$(3) \qquad \|f_{n+1} - f_n\| < C\lambda^n$$

In numerical experiments, this is invariably the case. Examples are constructed in [11] to show that the least λ for which (3) is true may be arbitrarily close to 1. The best convergence estimate known is that of Diliberto and Straus:

$$\|f_{n+1} - f_n\| < C/\log n$$

Special results in the discrete case of this problem have been developed in [11] and [15]. For example, the set of all best approximations to a given f has been characterized, and the conditions for a unique best approximation have been found.

If the function f being approximated is defined on a subset of a Cartesian product, the theory is quite different. Existence of best approximations may fail and the Diliberto-Straus Algorithm may not work. Reference [8] is pertinent here.

If it is desired to approximate f(x,y) by g(x)+h(y) with g and h in prescribed subspaces of C(X) and C(Y), then the theory again is different. The existence of best approximations, for one thing, seems to depend on additional assumptions. Work on this aspect of the problem is in progress by the author and John Respess.

4. THE L-1 APPROXIMATION PROBLEM.

For representing empirical functions that may be contaminated with errors, approximation schemes other than the uniform type are called for. In particular, L-1 and L-2 methods play important roles, the former being especially recommended when "outliers" are present in the data.

The L-1 theory offers an interesting challenge. The beginnings are contained in [16], but much remains to be done. In particular, satisfactory algorithms need to be developed. Two cases in which the existence of best L-1 approximations has been established are these:

I. The function f being approximated is in $L_1(X \times Y)$ and $g \in L_1(X)$, $h \in L_1(Y)$. It is assumed that X and Y have finite measure.

II. The function being approximated is continuous, and g and h are continuous. Here it is assumed that X and Y are topological spaces with finite Borel measures.

At this time, there exists no algorithm comparable to that of Diliberto and Straus for L-1 approximation. Indeed, the analogue of their algorithm does not work. An example, given in [15], is as follows. Let X=Y=[-1,1]. Let f(x,y) equal xy in the first quadrant, -xy in the third, and 0 elsewhere. Then the best approximations of f by functions of x or y alone are 0. The Diliberto-Straus Algorithm therefor produces $f_n=f$ for all n. But 0 is not a best approximation of the form g(x)+h(y).

References

1. Aumann, G. [1959]. "Über Approximative Nomographie II", Bayer. Akad. Wiss. Math. Nat. Kl. S.B., 103-109, MR22 #6968.

2. Bank, R. [1979]. "An automatic scaling procedure for a d'Yakanov-Gunn Iteration Scheme", to appear, Linear Algebra and its Applications.

3. Bank, R. and D.J. Rose [1977]. "Marching algorithms for elliptic boundary value problems I: The constant coefficient case", SIAM J. on Numerical Analysis 14, 792-829.

4. Bank, R. [1977]. "Marching algorithms for elliptic boundary value problems II: The variable coefficient case", SIAM J. on Numerical Analysis 14, 950-970.

5. Buck, R.C. [1968]. "Alternation Theorems for Functions of Several Variables", J. of Approximation Theory 1, 325-334.

6. Buck, R.C. [1975]. "Approximate Functional Complexity", Bull. Amer. Math. Soc., 81, 1112-1114, MR52 #8356.

7. Buck, R C. [1973]. "Approximation Theory and Functional Equations II", J. Approximation Theory 9, 121-125.

8. Buck, R.C. [1972]. "On Approximation Theory and Functional Equations", J. Approximation Theory 5, 228-237, MR51 #13535.

9. Diliberto, S.P. and E.G. Straus [1951]. "On the approximation of a function of several variables by the sum of functions of fewer variables", Pacific J. Math., Vol. 1, #2, 195-210.

10. Fulkerson, D.R. and P. Wolfe [1962]. "An Algorithm for Scaling Matrices", SIAM Review 4, #2, 142-146.

11. von Golitschek, M. and E.W. Cheney [1978]. "On the algorithm of Diliberto and Straus for approximating bivariate functions by univariate ones", Center for Numerical Analysis, The University of Texas, Report #141.

12. Gordon, W.F. and E.W. Cheney [1978]. "Bivariate and multivariate interpolation with noncommutative projectors", Linear Spaces and Approximation, 381-387, P.L. Butzer and B. Sz. Nagy eds., ISNM Vol. 40, Birkhauser Verlag.

13. Hitotumatu, Sin [1968]. "On the Approximation of a Function of Two Variables by the Product", Information Processing in Japan 8, 41-43, MR41 #4081.

14. Kelley, C.T. [1978]. "A note on the approximation of functions of several variables by sums of functions of one variable", Math. Research Center, Madison, Report #1873.

15. Light, W.A. and E.W. Cheney [1978]. "On the approximation of a bivariate function by the sum of univariate functions", Center for Numerical Analysis, The University of Texas, Report #140.

16. Light, W.A., J.H. McCabe, G.M. Phillips and E.W. Cheney [1979]. "The approximation of bivariate functions by sums of univariate ones using the L-1 metric", Center for Numerical Analysis, The University of Texas, Report #147.

17. Lorentz, G.G. [1976]. "The 13-th problem of Hilbert", Proceedings of Symposia in Pure Mathematics, Amer. Math. Soc., 28, 419.

18. Richter-Dyn, N. [1978]. "A Straightforward generalization of Diliberto and Straus' algorithm does not work", Math. Research Center, Madison, Wisconsin, December 1978.

Center for Numerical Analysis
The University of Texas at Austin
Austin, Texas 78712

ONE-PASS CURVE FITTING*

Philip W. Smith
Department of Mathematics
Texas A&M University
College Station, Texas 77843

ABSTRACT. We propose a method to reduce data which requires minimal storage and can be implemented in real time.

1. INTRODUCTION. Consider the following situation: A machine takes measurements at the rate of 300 samples per second producing data of the form $(i\Delta, y_i)$ until the machine is turned off. How can we efficiently process this data? We must assume that the measurements y_i are not exact and hence must be smoothed. This suggests least squares smoothing or perhaps digital filtering. However if the machine is running for any length of time the number of data points will be huge (e.g. 90,000 for five minutes). With this much data typical least squares smoothing will be quite costly. On the other hand, digital filtering is generally easy to perform but bad data must be recognized and dealt with. What is needed is a robust method which will essentially parameterize this data in real time.

In this paper we propose a possible solution to the above problem. The idea is to process the data as it becomes available and to produce both the knots and linear coefficients of a spline which will approximate the data. As will be indicated in the second section, the theory of spline least squares approximation is just now beginning to become understood to the point which makes the algorithms feasible. The third section contains a technical description of the algorithm and in the fourth section we display some numerical results.

2. LEAST SQUARES APPROXIMATION BY SPLINES. In this section we will present the theoretical underpinnings for our one-pass curve fitting algorithm. These results are contained in two papers of Barrow and Smith [1, 2].

To expedite what follows some notation will now be introduced. Let t be a mapping from R onto R satisfying

(2.1) $t \in C^1(R)$

(2.2) $t' > 0$

(2.3) $t(0) = 0$ and $t(1) = 1$.

We will use the function t to distribute (in a regular but possibly nonuniform manner) knots on the interval $[0,1]$. Let N and k be

*This research was supported by the U.S. Army Research Office under Grant No. DAHC 04-75-G-0186.

positive integers and let

$$\underline{t} := t\left(\frac{1-k}{N}\right) < \dots < t(0) = 0 < \dots < t\left(\frac{N-1}{N}\right) < t(1) = 1$$

$$< \dots < t\left(\frac{N+k-1}{N}\right).$$

$$\left(= t_{1-k} < \dots < t_{N+k-1}\right).$$

We will define $S_N^k(t)$ to be the linear space of splines of order k with knot sequence $\underline{t}$. That is

$$S_N^k(t) := \operatorname{span}\{N_{i,k} : i = 1-k, \dots, N-1\}$$

where as usual (see [3])

$$N_{i,k}(t) := (t_{i+k} - t_i)[t_i, \dots, t_{i+k}]_s (s-t)_+^{k-1} .$$

Let $P_N(t)$ be the least squares projector onto $S_N^k(t)$. That is, $P_N(t)$ is the unique linear operator mapping

$$L_2[0,1] \to S_N^k(t)$$

satisfying for all $s \in S_N^k(t)$.

$$\text{(2.4)} \qquad \int_0^1 \left(f - P_N(t)f\right)(x)s(x)dx = 0.$$

With these notations we are prepared to state the two main theorems from [1]. Let

$$\|f\| := \left(\int_0^1 f^2(x)dx\right)^{1/2} .$$

Then we have:

Theorem 1. For $f \in C^k[0,1]$ we have

$$\lim_{N\to\infty} N^k \|f - P_N(t)f\| = C_k \left(\int_0^1 |f^{(k)}(t(x))|^2 (t'(x))^{2k+1} dx\right)^{1/2}$$

where $C_k = (|B_{2k}|/2k!)$ and B_{2k} is the 2k-th Bernoulli number.

If we optimize the knot locations and let $N \to \infty$ we obtain

Theorem 2. If $f \in C^k[0,1]$ we have

$$\lim_{N\to\infty} N^k \inf\{\|f - P_N(t)f\|: t \text{ as above}\} = C_k\left(\int_0^1 |f^{(k)}(x)|^\sigma\right)^{1/\sigma}$$

where C_k is as above and $\sigma = (k + 1/2)$.

These two theorems and their proofs indicated that (asymptotically) L_2 approximation by splines is a local phenomenon. In [2] we attempted to produce good L_2 approximations from local information. That is, we wished to find linear functionals λ_i so that $\text{supp}(\lambda_i) \subset \text{supp } N_{i,k} = [t_i, t_{i+k}]$ and satisfying

$$\lim_{N\to\infty} N^k\left(\left\|f - \sum_{i=1-k}^{N-1} \lambda_i(f)N_{i,k}\right\| - \|f - P_N(t)f\|\right) = 0$$

for $f \in C^k[0,1]$. In particular, the λ_i for $k = 2, 3,$ and 4 are listed below $(h_i := t_{i+1} - t_i)$

(2.5) $\quad k = 2, \quad \lambda_{i-1}(f) := f(t_i) - f''(t_i)h_i^2/12$

(2.6) $\quad k = 3, \quad \lambda_{i-1}(f) := f(t_i) + h_i f'(t_i)/2 - \dfrac{f^{(3)}(t_i)h_i^3}{24}$

(2.7) $\quad k = 4, \quad \lambda_{i-2}(f) := f(t_i) + (h_i - h_{i-1})f'(t_i)/3 - h_i h_{i-1} f^{(2)}(t_i)/6$
$\qquad\qquad + f^{(4)}(t_i)h_i^3(h_i/30 + h_{i-1}/3)/24.$

Thus if we are given the knot sequence $\underline{t}$ and the information $(f(t_i), \ldots, f^{(k)}(t_i))$ we can produce a good L_2 approximation to f from $S_N^k(t)$. Also note that if we are given $(f(t_i), \ldots, f^{(k)}(t_i))$ for $i = 1-k, \ldots, L < N$ then on the interval $[0, (L-1)/N]$ for $k = 2$ and 3 we can produce a good L_2 approximation to f independent of the future information by

$$\sum_{j=1-k}^{N-1} \lambda_j(f)N_{j,k} \; .$$

The importance of this observation should not be underestimated. In essence, what we have produced is (asymptotically) the best $L_2[0,1]$ approximation to f on $[0, L-1/N]$ from $S_N^k(t)$ without any quantitative information about f on $[L/N, 1]$.

In order to form the sum

$$\sum_{j=1-k}^{L-1} \lambda_j(f)N_{j,k}$$

it is necessary to know only $\{t_{1-k}, \ldots, t_{L+k-1}\}$. Thus, if there is a method for adaptively selecting the knots we can incorporate this method into our algorithm. It is known from [1] that an asymptotically optimal procedure for selecting knots is given by choosing $\underline{t}$ so that

$$\int_{t_{i-1}}^{t_i} |f^{(k)}(\tau)|^\sigma d\tau = \left(\int_0^1 |f^{(k)}(\tau)|^\sigma d\tau\right) \Big/ N.$$

With this selection one obtains the error rate indicated in Theorem 2. In the next section we will show how to take advantage of these results to produce a one-pass curve fitting algorithm.

3. A ONE-PASS CURVE FITTING ALGORITHM. Let us assume that we have data of the form

$$\{(i\Delta, y_i)\}_{i=0}^{L}$$

where Δ is a positive constant, $y_i = f(i\Delta) + \varepsilon_i$, $f \in C^k$, and the ε_i are random uncorrelated variables with mean zero and unknown variance. The idea is to produce a spline approximation to this data in real time.

There are two parts to the algorithm, namely knot placement and computation of the corresponding B-spline coefficient. The user inputs are:

(3.1) k, the order of the spline approximation,

(3.2) Δ, the sample spacing,

(3.3) M, an integer $> k$ (a smoothing parameter),

(3.4) $\varepsilon > 0$, the error to be tolerated on (t_i, t_{i+1}).

Let π_i denote the best least square approximation to the data

$$\{(j\Delta, y_j)\}_{j=i}^{i+M}$$

from polynomials of degree less than or equal to k. The algorithm proceeds as follows:

1. Set $t_1 = \ldots = t_k = 0$, $i_0 = 0$, and $r = k + 1$
2. Find the smallest integer $i \leq L/M$ so that $C_k\left(\sum_{j=i_0}^{i} |(\pi_{jM})^{(k)}|^\sigma M\Delta\right)^{1/\sigma} \geq \varepsilon$. If no such i exists set $t_r = L\Delta$ and go to 7.
3. If $r = k+1$ go to 6. Otherwise set $t_r = (i+1)M\Delta$.

4. Compute

$$[\pi_{iM}(t_r), \pi^{(1)}_{iM}(t_r), \ldots, \pi^{(k)}_{iM}(t_r)]$$

and use these values as estimates for $[f(t_r), \ldots, f^{(k)}(t_r)]$ in formulas (2.5), (2.6), (2.7), producing $A(r-2)$ the coefficient for $N_{r-2,k}$ for $k = 2, 3$ or if $k = 4$ the coefficient $A(r-3)$ for $N_{r-3,k}$.

5. Set $r = r+1$, $i_0 = i+1$, and go to 2.

6. Use π_0 to compute $[\pi_0(0), \ldots, \pi_0^{(k)}(0)]$ and choose $A(1),\ldots,A(\ell)$, $\ell = 1, [k/2]$, so that the spline s satisfies $s^{(m)}(0) = \pi_0^{(m)}(0)$ for $m = 1, [k/2]$. Set $r = r+1$ and $i_0 = i+1$ and go to 2.

7. Use π_{L-M} to obtain $A(r-2)$ or $A(r-3)$ as in 4. Then set $t_{r+1} = \ldots = t_{r+k-1} = L$ and choose $A(r-1)$ (if $k = 2, 3$) or $A(r-1)$ and $A(r-2)$ if $k = 4$ so that

$$s(t) = \sum_{j=1}^{r-1} A(j)N_{j,k}(t)$$

satisfies $s(L\Delta) = \pi_{L-M}(L\Delta)$ if $k = 2, 3$ or $s^{(\ell)}(L\Delta) = \pi^{(\ell)}_{L-M}(L\Delta)$ for $\ell = 0, 1$ if $k = 4$, and stop.

Some comments should be made concerning this algorithm. In step 2 the left side of the inequality represents (from Theorem 2) an estimate of the L_2 norm of the error between f and the spline approximation if t_r is placed at $(i+1)M\Delta$. Thus the user specifies the error ε, he is willing to tolerate. We could have tried to locate t_r more accurately relative to this criteria at the expense of making the algorithm more complex. What is done in steps 6 and 7 is not very crucial to the method. These steps just take care of the first and last few intervals (which should be insignificant relative to the total curve) and could certainly be handled differently.

The computation of $\pi^{(k)}_{jM}$ is accomplished very efficiently by precalculating the k-th orthogonal polyonmial on $M+1$ equally spaced points and then taking the innerproduct with the data. Notice that if relatively few knots are placed per data point then this algorithm processes the data with just over one multiplication and one addition per data point.

Probably the most subjective parameter for the user to specify is M. The algorithm guarantees that the knots will be at least $M\Delta$ apart. Thus one would not want M to be too large. Roughly speaking, the user chooses M large enough so that π_{jM} "filters" most of the noise but small enough so that π_{jM} can faithfully represent the function f.

4. NUMERICAL EXAMPLES. In this section we want to illustrate how the method behaves numerically. We have chosen a function to approximate which is relatively flat followed by a sharp peak and then relatively flat again followed by two peaks. Such functions tend to give ordinary curve fitting routines much difficulty. The function is

$$f(x) = (x-3)^3/\left([(x-1)^2 + .001][(x-5)^2 + .005][(x-6)^2 + .001]\right).$$

It was decided to use exact data (i.e. no noise) and $M = 5$ with $\Delta = .001$. We started at $x = 0$ and used 10,000 data points (at step Δ). We specified ε to be .001 and then computed the error on each subinterval. Two tables follow with results for quadratic and cubic splines.

We have listed the results in the following two tables. The first column contains the number of the knot, the second column contains the knots computed by the algorithm, the third contains the spline coefficients, and the fourth contains the L_2 error on the interval (t_i, t_{i+1}) as computed by a five point Gauss quadratic formula. The reader should note that it takes many more knots for the quadratic spline fit to match the cubic spline fit. Also note that around 1, 5, and 6 many more knots are placed because here the k-th derivative is large. Finally, the reader should be aware that the errors are all larger than .001 because the algorithm always chooses the knots in such a manner that the predicted error is larger than .001.

REFERENCES

1. D. L. Barrow and P. W. Smith, Asymptotic properties of best $L_2[0,1]$ approximation by splines with variable knots. Quart. of Appl. Math., (Oct. 1978), 293-304.

2. D. L. Barrow and P. W. Smith, Efficient L_2 approximation by splines, manuscript.

3. C. de Boor, On calculating with B-splines, J. Approx. Theory, 6 (1972), 50-62.

TABLE 1. QUADRATIC SPLINE

I	Knots t(I)	Coef A(I)	L_2 Error On $\left[t(I),t(I+1)\right)$	I	Knots t(I)	Coef A(I)	L_2 Error On $\left[t(I),t(I+1)\right)$
1	.00000	-.29963(-1)		51	.53400(1)	.11633(2)	.54 (-2)
2	.00000	-.39743(-1)		52	.54250(1)	.13352(2)	.92 (-2)
3	.00000	-.11980	.25 (-2)	53	.55450(1)	.15735(2)	.30 (-2)
4	.47000	-.30244	.30 (-2)	54	.56150(1)	.18945(2)	.19 (-2)
5	.68500	-.65934	.28 (-2)	55	.56700(1)	.23225(2)	.16 (-2)
6	.79500	-.13070(1)	.30 (-2)	56	.57150(1)	.28723(2)	.28 (-2)
7	.86000	-.24031(1)	.25 (-2)	57	.57550(1)	.35032(2)	.19 (-2)
8	.90000	-.43141(1)	.37 (-2)	58	.57850(1)	.43166(2)	.12 (-2)
9	.93000	-.77959(1)	.13 (-2)	59	.58100(1)	.54013(2)	.33 (-2)
10	.95000	-.15373(2)	.49 (-2)	60	.58350(1)	.66100(2)	.32 (-2)
11	.97500	-.21098(2)	.11 (-1)	61	.58550(1)	.80070(2)	.15 (-2)
12	.99000	-.18378(2)	.73 (-2)	62	.58700(1)	.99417(2)	.20 (-2)
13	.10050(1)	-.12740(2)	.17 (-2)	63	.58850(1)	.12264(3)	.59 (-2)
14	.10150(1)	-.69170(1)	.14 (-1)	64	.59000(1)	.14701(3)	.25 (-2)
15	.10300(1)	-.40074(1)	.15 (-1)	65	.59100(1)	.17949(3)	.22 (-2)
16	.10500(1)	-.21921(1)	.39 (-2)	66	.59200(1)	.22380(3)	.35 (-2)
17	.10700(1)	-.11485(1)	.79 (-2)	67	.59300(1)	.28572(3)	.54 (-2)
18	.11000(1)	-.55907	.77 (-2)	68	.59400(1)	.37404(3)	.78 (-2)
19	.11400(1)	-.24259	.10 (-1)	69	.59500(1)	.50112(3)	.87 (-2)
20	.12050(1)	-.96408(-1)	.13 (-1)	70	.59600(1)	.63458(3)	.81 (-2)
21	.13200(1)	-.51077(-1)	.19 (-1)	71	.59700(1)	.73541(3)	.36 (-2)
22	.15450(1)	-.16126(-1)	.30 (-1)	72	.59750(1)	.84339(3)	.69 (-2)
23	.20950(1)	.51771(-3)	.25 (-1)	73	.59800(1)	.94815(3)	.10 (-1)
24	.34000(1)	.48256(-1)	.30 (-2)	74	.59850(1)	.10329(4)	.10 (-1)
25	.40150(1)	.20951	.26 (-2)	75	.59900(1)	.10784(4)	.56 (-2)
26	.43050(1)	.56723	.25 (-2)	76	.59950(1)	.10710(4)	.12 (-1)
27	.44800(1)	.12097(1)	.24 (-2)	77	.60000(1)	.10116(4)	.21 (-1)
28	.45950(1)	.22358(1)	.21 (-2)	78	.60050(1)	.91557(3)	.16 (-1)
29	.46750(1)	.38430(1)	.18 (-2)	79	.60100(1)	.80297(3)	.64 (-2)
30	.47350(1)	.63841(1)	.26 (-2)	80	.60150(1)	.69039(3)	.76 (-2)
31	.47850(1)	.10006(2)	.31 (-2)	81	.60200(1)	.53443(3)	.23 (-1)
32	.48250(1)	.14738(2)	.23 (-2)	82	.60250(1)	.38524(3)	.97 (-1)
33	.48550(1)	.21506(2)	.14 (-2)	83	.60350(1)	.28296(3)	.38 (-1)
34	.48800(1)	.32370(2)	.22 (-2)	84	.60450(1)	.21295(3)	.55 (-2)
35	.49050(1)	.49709(2)	.25 (-2)	85	.60550(1)	.16421(3)	.70 (-2)
36	.49300(1)	.69620(2)	.98 (-2)	86	.60650(1)	.12947(3)	.59 (-2)
37	.49550(1)	.85197(2)	.47 (-2)	87	.60750(1)	.10410(3)	.39 (-2)
38	.49700(1)	.97573(2)	.42 (-2)	88	.60850(1)	.80577(2)	.43 (-2)
39	.49850(1)	.10306(3)	.29 (-2)	89	.60950(1)	.61634(2)	.91 (-2)
40	.50000(1)	.94861(2)	.23 (-1)	90	.61100(1)	.48392(2)	.40 (-2)
41	.50200(1)	.83548(2)	.77 (-2)	91	.61250(1)	.37287(2)	.27 (-2)
42	.50350(1)	.69336(2)	.30 (-2)	92	.61400(1)	.27683(2)	.49 (-2)
43	.50500(1)	.51973(2)	.10 (-1)	93	.61600(1)	.20756(2)	.51 (-2)
44	.50700(1)	.39579(2)	.17 (-1)	94	.61850(1)	.15581(2)	.23 (-2)
45	.51000(1)	.30983(2)	.44 (-2)	95	.62100(1)	.11242(2)	.29 (-2)
46	.51250(1)	.24397(2)	.26 (-2)	96	.62400(1)	.78108(1)	.41 (-2)
47	.51550(1)	.19611(2)	.33 (-2)	97	.62800(1)		
48	.51900(1)	.16070(2)	.33 (-2)	98	.63300(1)		
49	.52300(1)	.13600(2)	.38 (-2)	99	.63900(1)		
50	.52800(1)	.11992(2)	.37 (-2)				

TABLE 2. CUBIC SPLINE

I	Knots t(I)	Coef A(I)	L_2 Error On (t(I),t(I+1))
1	.00000	-.29963(-1)	
2	.00000	-.39230(-1)	
3	.00000	-.37534(-1)	
4	.00000	-.16828	.53 (-2)
5	.52000	-.56136	.95 (-2)
6	.74000	-.15066(1)	.81 (-2)
7	.84500	-.37973(1)	.84 (-2)
8	.90500	-.10043(2)	.95 (-2)
9	.94500	-.21207(2)	.91 (-2)
10	.97500	-.18762(2)	.25 (-1)
11	.99500	-.73560(1)	.10 (-1)
12	.10100(1)	-.32365(1)	.23 (-1)
13	.10300(1)	-.13298(1)	.48 (-1)
14	.10600(1)	-.51305	.11 (-1)
15	.11000(1)	-.19431	.21 (-1)
16	.11600(1)	-.10203	.29 (-1)
17	.12700(1)	-.26453(-1)	.41 (-1)
18	.15000(1)	-.37728(-2)	.56 (-1)
19	.20900(1)	-.99345(-2)	.32 (-1)
20	.35100(1)	.12918	.86 (-2)
21	.41250(1)	.63094	.75 (-2)
22	.44250(1)	.18651(1)	.71 (-2)
23	.46000(1)	.43371(1)	.61 (-2)
24	.47100(1)	.89325(1)	.52 (-2)
25	.47850(1)	.17782(2)	.39 (-2)
26	.48400(1)	.34048(2)	.24 (-2)
27	.48850(1)	.61144(2)	.28 (-2)
28	.49300(1)	.88942(2)	.29 (-2)
29	.49550(1)	.10558(3)	.81 (-2)
30	.49800(1)	.94410(2)	.16 (-1)
31	.50050(1)	.71440(2)	.46 (-2)
32	.50300(1)	.46864(2)	.59 (-2)
33	.50550(1)	.30557(2)	.21 (-1)
34	.50850(1)	.20849(2)	.32 (-1)
35	.51350(1)	.15200(2)	.36 (-2)
36	.51900(1)	.12162(2)	.81 (-2)
37	.52550(1)	.11429(2)	.89 (-2)
38	.53450(1)	.12907(2)	.50 (-2)
39	.54650(1)	.16861(2)	.25 (-2)
40	.55850(1)	.23611(2)	.60 (-2)
41	.56800(1)	.33809(2)	.72 (-2)
42	.57500(1)	.48198(2)	.53 (-2)
43	.58000(1)	.69189(2)	.71 (-2)
44	.58400(1)	.98428(2)	.56 (-2)
45	.58700(1)	.13932(3)	.77 (-2)
46	.58950(1)	.19637(3)	.83 (-2)
47	.59150(1)	.27929(3)	.22 (-2)
48	.59300(1)	.40475(3)	.30 (-2)
49	.59450(1)	.58122(3)	.20 (-2)
50	.59600(1)	.78635(3)	.71 (-2)
51	.59700(1)	.10100(4)	.17 (-1)
52	.59800(1)	.11140(4)	.20 (-1)
53	.59900(1)	.98212(3)	.81 (-1)
54	.60000(1)	.74152(3)	.19 (-1)
55	.60100(1)	.53276(3)	.75 (-1)
56	.60200(1)	.35828(3)	.91 (-2)
57	.60300(1)	.23832(3)	.28 (-1)
58	.60400(1)	.16083(3)	.30 (-1)
59	.60550(1)	.10888(3)	.34 (-2)
60	.60700(1)	.72576(2)	.88 (-2)
61	.60850(1)	.47438(2)	.11 (-1)
62	.61050(1)	.30106(2)	.10 (-1)
63	.61300(1)	.18717(2)	.10 (-1)
64	.61600(1)	.11081(2)	.10 (-1)
65	.62000(1)	.62632(1)	.10 (-1)
66	.62500(1)	.32785(1)	.10 (-1)
67	.63200(1)	.15715(1)	.11 (-1)
68	.64150(1)	.67127	.13 (-1)
69	.65550(1)	.25049	.14 (-1)
70	.67650(1)		
71	.71000(1)		
72	.76800(1)		
73	.88200(1)		

EMPIRICAL STOPPING RULES FOR DATA FITTING BY SPLINES

Patricia L. Smith
Institute of Statistics

and

Philip W. Smith*
Department of Mathematics

Texas A&M University

1. INTRODUCTION. We consider the problem of recovering a smooth function, f, from a large number of discrete observations which have been contaminated by noise. In particular, we assume the model

$$(1.1)\quad y(x_i) = f(x_i) + \varepsilon(x_i),\quad 0 \le x_1 < \dots < x_n \le 1$$

where the noise variables $\{\varepsilon(x_i)\}_{i=1}^{n}$ are a random sample from a continuous zero mean distribution with unknown variance σ^2, and $f \in C^k[0,1]$.

Our approach to recovering (a good approximation to) f is based on the theoretical results outlined in section 2 of the previous paper [7]. We use these results to estimate the $L_2[0,1]$ error between f (which is unknown) and a spline function of order k which is obtained by finding the least squares approximation to the data from $S^k(\underline{t})$ (splines of order k with knot sequence $\underline{t}$). Of course, the problem is to choose the correct knot sequence $\underline{t}$ since too many knots will result in overfit (i.e. fitting the noise) whereas too few knots will result in underfit (i.e. failure to fit the function at all). In this paper we limit the discussion to procedures for determining the proper number of equally spaced knots which will produce a good approximation to f. More general results will be presented at a later date. The techniques we employ here should be contrasted with those of Wahba and Wold [8] and Agarwal and Studden [1, 2] who also consider estimation of f in the model (1.1).

Section 2 deals with the technicalities and notation involved with least squares approximation by splines along with the requisite theory of L_2 approximation by splines. In section 3 we discuss the statistical aspects of this problem and in section 4 we present some numerical examples and concluding remarks.

2. LEAST SQUARES APPROXIMATION BY SPLINES. We will use the same notation as in section 2 of [7], the preceding paper in this volume. Thus t will denote a knot distribution function and $S_N^k(t)$ the corresponding subspace of splines of order k with interior knots at $0 < t(1/N) < \dots < t\big((N-1)/N\big) < 1$. That is, $S_N^k(t) := \operatorname{span}\{N_{i,k} : i = 1-k,\dots,N-1\}$ and $P_N(t)$ is the corresponding L_2 projection.

*This author was supported by the U.S. Army Research Office under Grant No. DAHC 04-75-G-0186.

Setting $\underline{N}^T(x) := \left(N_{1-k,k}(x), \ldots, N_{N-1,k}(x)\right)$ we see that

$$(2.1) \qquad \left(P_N(t)f\right)(x) = \underline{\alpha}^T\underline{N}(x)$$

where $\underline{\alpha}$ satisfies the normal equations

$$(2.2) \qquad A\underline{\alpha} = \underline{b}$$

where

$$(2.3) \qquad A = \int_0^1 \underline{N}(x)\underline{N}^T(x)dx$$

and

$$(2.4) \qquad \underline{b} = \int_0^1 \underline{N}(x)f(x)dx \ .$$

In the special case $t(x) = x$ (i.e. equally spaced knots) we see that [7, Theorem 1]

$$(2.5) \qquad \int_0^1 \left(f - P_N(t)f\right)^2(x)dx = N^{-2k}C_k^2\int_0^1 |f^{(k)}(x)|^2dx + o\left(N^{-2k}\right)$$

Suppose we are given discrete data $\{(x_i,y_i)\}_{i=1}^n$ where $x_i = (i-1)/n$. Then we can produce the best least squares approximation to the data from $S_N^k(t)$ by solving

$$(2.6) \qquad \tilde{A}\underline{\tilde{\alpha}} = \underline{\tilde{b}}$$

where

$$(2.7) \qquad \tilde{A} = \frac{1}{n}\sum_{j=1}^n \underline{N}(x_j)\underline{N}^T(x_j)$$

and

$$(2.8) \qquad \underline{\tilde{b}} = \frac{1}{n}\sum_{j=1}^n y_j\underline{N}(x_j) \ .$$

Notice that the entries in $\tilde{A}$ are Riemann sums approximating the corresponding entries in A and the entries in $\underline{\tilde{b}}$ would correspond to the entries in $\underline{b}$ provided $y_i = f(x_i) + \varepsilon(x_i)$ as in (1.1). Thus for n large $\tilde{A}$ is nearly A and $\underline{\tilde{b}}$ is nearly $\underline{b}$.

We want to develop a method for approximating the error

$$\int_0^1 \left(f(x) - s(x)\right)^2dx$$

where $s \in S_N^k(t)$ and $s(x) := \underline{\tilde{\alpha}}^T\underline{N}(x)$ and $\underline{\tilde{\alpha}}$ is determined by (2.6) - (2.8).

Using (2.5) we see that this error is (asymptotically)

$$N^{-2k}C_k^2\int_0^1 |f^{(k)}(x)|^2 dx.$$

Now we know N and C_k but we don't know f (or $f^{(k)}$). We do know s, however, and s approximates f so that $s^{(k)}$ should approximate $f^{(k)}$. The only problem with this reasoning is that $s^{(k)}$ is not well defined. Recall that $s \in C^{k-2}[0,1]$ and does not have a classically defined k-th derivative. However, as has been indicated by de Boor [5] and Dodson [6] one can construct a function which approximates $f^{(k)}$ as follows: Compute $s^{(k-1)}$ which will be a piecewise constant function

$$(2.9) \qquad s^{(k-1)}(x) = \{r_i \text{ if } i/N \leq x < (i+1)/N\}.$$

Then smooth this function by defining $p(x)$ to be the unique piecewise linear continuous function which interpolates $s^{(k-1)}$ at the points $\left(i + (1/2)\right)/N$, $i = 0, \ldots, N-1$ with knots at $\left(i + (1/2)\right)/N$ $i = 1, \ldots, N-2$. Then p' will be a piecewise constant function which approximates $f^{(k)}$. Thus we define

$$(2.10) \qquad \text{ATE}(N) := N^{-2k}C_k^2\int_0^1 |p'(x)|^2 dx.$$

Now ATE(N) approximates the square of the L_2 error between f and the subspace $S_N^k(t)$.

In order to get a feeling for whether the asymptotics have taken over and for whether we are fitting the noise we not only monitor ATE but we also compute

$$\text{FACTOR}(N) := \frac{\log\left(\text{ATE}(N-1)/\text{ATE}(N)\right)}{\log\left((N-1)/N\right)}.$$

This number should approach (-2k) as N gets large provided the noise is not affecting the approximation.

3. STATISTICAL AND EMPIRICAL ASPECTS. For a point $\xi \in [0,1]$, the bias of the prediction at ξ using the least squares spline fit (from $S_N^k(t)$) $\underline{\tilde{\alpha}}^T N = \widehat{P_N}f$ is given by

$$(3.1) \qquad f(\xi) - E\,\widehat{P_N}f(\xi)$$

where E denotes expectation. Using the results of Section 2 in the previous paper Agarwal and Studden [1] have shown that the integrated squared bias B of a prediction

$$B := \int_0^1 \left(f(x) - E\widehat{P_N}f(x)\right)^2 dx$$

has the asymptotic value (as $N \to \infty$) equal to the left hand side of (2.5). A good estimate of $f^{(k)}$ in the right side of (2.5) will thus help provide a good estimate of B. Our estimate of B is ATE which has a dual purpose. First we use it to estimate B, which will be an average squared bias since the integration is over $[0,1]$. Secondly, successive values of ATE are used to determine the correct degree of smoothing (i.e. via FACTOR).

The procedure we follow is to compute $\hat{P_N}f$, ATE(N), FACTOR(N) for $N = 1, 2, \ldots$. Since ATE(N) is computed from $\hat{P_N}f$ it will generally happen that (for small N) $\hat{P_N}f \cong P_N(t)f$. Thus we expect to see the following phenomenon. ATE will begin to decrease as N increases and will approach the asymptotic rate N^{-2k}. Thus FACTOR will be negative and begin to decrease to $-2k$. However at some point the estimators $\hat{P_N}f$ will begin to fit the noise. At this point our estimate of $f^{(k)}$ will not be very good and FACTOR will probably fluctuate wildly (in many cases even becoming positive). Ideally, we would hope to observe at least two consecutive values of FACTOR near $-2k$. Then we would believe that the asymptotics have taken over and the estimate ATE of the integrated squared bias is accurate. In practice we have found it useful to cross-reference results from splines of different order.

4. NUMERICAL EXPERIMENTS. In this section we present some numerical experiments which we can use to see how the theory presented in the previous section applies to a specific example. We will be trying to recover the function

$$f(x) = 4.26\left(e^{-3.25x} - 4e^{-6.5x} + 3e^{-9.75x}\right) .$$

Two tables are presented below. The first table shows how linear, quadratic and cubic splines behave when there is no noise ($\varepsilon(x_i) = 0$). In the second table we have used a normal random number generator to produce the $\{\varepsilon(x_i)\}_{i=1}^{600}$ with a sample variance of .039. We have used 600 equally spaced sample points

$$\{x_i = (i-1)/600\}_{i=1}^{600}$$

on the unit interval. The data we have is

$$\left\{\left(x_i,\ y_i \equiv \left(f(x_i) + \varepsilon(x_i)\right)\right)\right\}_{i=1}^{600} .$$

In the tables we report the following numbers for linear, quadratic, and cubic splines, with the headings

(4.1) # Intervals is the number of interior knots + 1

(4.2) MSE is $\frac{1}{600}\sum_{i=1}^{600}\left(y_i - \hat{P_N}f\right)^2$

where $\hat{P_N f}$ is the spline with N interior knots which is the least squares best approximation to the data,

(4.3) ATE and FACTOR are as in section 2.

One should notice in Table 1 that as N increases ATE is generally a better approximator of MSE as is to be expected. It is also important to note the behavior of FACTOR. We expect FACTOR to eventually become -2k where k is the order of the spline. In this case as in most cases which we have run FACTOR is converging monotonely to -2k from above.

Now compare the behavior of FACTOR in Table 2. One can see the effects of overfit by examining rows 13 through 15. If we were not overfitting we would expect these numbers to be negative and decreasing to -2k. There are several plausible methods for choosing the spline which best fits f. One method would be to choose the middlespline from the longest sequence of decreasing negative numbers. Another would be to look for three successive numbers nearest -2k. We have rounded all number to two digits.

TABLE 1. (NO NOISE)

	Linear			Quadratic			Cubic		
# Intervals	MSE	ATE	FACTOR	MSE	ATE	FACTOR	MSE	ATE	FACTOR
2	.23 (-1)	.31 (-2)		.29 (-1)	.48 (-4)		.13 (-1)	.79 (-3)	
3	.27 (-1)	.55 (-3)	-4.3	.15 (-1)	.36 (-3)	5	.31 (-2)	.22 (-3)	-3.0
4	.21 (-1)	.40 (-3)	-1.1	.56 (-2)	.31 (-3)	-4.7	.57 (-3)	.94 (-4)	-3.0
5	.14 (-1)	.63 (-3)	2.0	.20 (-2)	.23 (-3)	-1.4	.12 (-3)	.40 (-4)	-3.8
6	.83 (-2)	.71 (-3)	.6	.79 (-3)	.14 (-3)	-2.4	.30 (-4)	.17 (-4)	-4.7
7	.51 (-2)	.67 (-3)	- .4	.34 (-3)	.91 (-4)	-3.1	.97 (-5)	.75 (-5)	-5.3
8	.33 (-2)	.58 (-3)	-1.1	.16 (-3)	.56 (-4)	-3.6	.37 (-5)	.35 (-5)	-5.7
9	.21 (-2)	.48 (-3)	-1.6	.80 (-4)	.35 (-4)	-3.9	.16 (-5)	.17 (-5)	-6.1
10	.15 (-2)	.39 (-3)	-1.9	.44 (-4)	.23 (-4)	-4.2	.78 (-6)	.88 (-6)	-6.3
11	.10 (-2)	.31 (-3)	-2.2	.25 (-4)	.15 (-4)	-4.5	.41 (-6)	.47 (-6)	-6.5
12	.73 (-3)	.25 (-3)	-2.4	.15 (-4)	.99 (-5)	-4.6	.22 (-6)	.26 (-6)	-6.7
13	.53 (-3)	.21 (-3)	-2.6	.96 (-5)	.67 (-5)	-4.8	.13 (-6)	.15 (-6)	-6.8
14	.40 (-3)	.17 (-3)	-2.7	.63 (-5)	.47 (-5)	-4.9	.76 (-7)	.92 (-7)	-6.9
15	.30 (-3)	.14 (-3)	-2.9	.42 (-5)	.33 (-5)	-5.0	.47 (-7)	.57 (-7)	-7.0

TABLE 2. $\left(y_i = f(x_i) + \varepsilon(x_i),\right.$ sample variance $\left. .039\right)$

	Linear			Quadratic			Cubic		
# Intervals	MSE	ATE	FACTOR	MSE	ATE	FACTOR	MSE	ATE	FACTOR
2	.62 (-1)	.28 (-2)		.67 (-1)	.45 (-4)		.52 (-1)	.73 (-3)	
3	.65 (-1)	.49 (-3)	-4.0	.55 (-1)	.32 (-3)	4.9	.42 (-1)	.21 (-3)	-3.0
4	.60 (-1)	.32 (-3)	-1.5	.45 (-1)	.30 (-3)	- .3	.39 (-1)	.11 (-3)	-2.0
5	.53 (-1)	.60 (-3)	2.9	.41 (-1)	.25 (-3)	- .7	.39 (-1)	.42 (-4)	-4.4
6	.47 (-1)	.72 (-3)	1.0	.39 (-1)	.15 (-3)	-2.8	.38 (-1)	.16 (-4)	-5.3
7	.43 (-1)	.67 (-3)	- .5	.39 (-1)	.94 (-4)	-3.1	.38 (-1)	.10 (-4)	-3.0
8	.42 (-1)	.59 (-3)	- .98	.38 (-1)	.65 (-4)	-2.8	.38 (-1)	.46 (-5)	-6
9	.40 (-1)	.50 (-3)	-1.3	.38 (-1)	.41 (-4)	-3.9	.38 (-1)	.16 (-5)	-8.8
10	.40 (-1)	.42 (-3)	-1.7	.38 (-1)	.25 (-4)	-4.8	.38 (-1)	.75 (-6)	-7.2
11	.39 (-1)	.34 (-3)	-2.2	.38 (-1)	.15 (-4)	-5.1	.38 (-1)	.60 (-6)	-2.3
12	.39 (-1)	.27 (-3)	-2.6	.38 (-1)	.10 (-4)	-4.3	.38 (-1)	.58 (-6)	- .35
13	.38 (-1)	.22 (-3)	-2.4	.38 (-1)	.93 (-5)	-1.6	.38 (-1)	.18 (-6)	-1.4
14	.38 (-1)	.19 (-3)	-2.6	.38 (-1)	.76 (-5)	-2.7	.38 (-1)	.28 (-6)	5.8
15	.38 (-1)	.16 (-3)	-1.6	.38 (-1)	.43 (-5)	-8.3	.38 (-1)	.16 (-5)	25.0

REFERENCES

1. G. G. Agarwal and W. J. Studden, Asymptotic integrated mean square error using polynomial splines. Mimeograph Series # 78-20, Department of Statistics, Purdue University, 1978.

2. __________, An algorithm for selection of design and knots in the response curve estimation by spline functions. Manuscript.

3. D. L. Barrow and P. W. Smith, Asymptotic properties of best $L_2[0,1]$ approximation by splines with variable knots. Quart. of Appl. Math. (Oct. 1978), 293-304.

4. __________, Efficient L_2 approximation by splines. Manuscript.

5. C. de Boor, Good approximation by splines with variable knots, II. In Conference on the Numerical Solution of Differential Equations, Dundee, 1973, Springer Lecture Notes, Vol. 363, 1974, 12-20.

6. D. S. Dodson, Optimal order approximation by polynomial spline functions. Ph.D. thesis, Purdue Univ., Lafayette, Indiana, 1972.

7. P. W. Smith, One-pass curve fitting. This volume.

8. G. Wahba and S. Wold. A completely automatic french curve: Fitting spline functions by cross validation. Communications in Statistics, 4 (1), 1-17, 1975.

NONPARAMETRIC DENSITY ESTIMATION: WHERE DO WE GO FROM HERE?[1]

R.A. Tapia and J.R. Thompson
Mathematical Sciences Department
Rice University
Houston, Texas 77001

ABSTRACT. The estimation of multidimensional probability density functions is an important unsolved problem in statistical methods. In this paper we review the state of the art with respect to kernel density estimation and maximum penalized likelihood density estimation in one dimension. This review is made with the purpose of motivating and suggesting several promising strategies for extending these successful approaches from one dimension to higher dimensions in an efficient manner.

1. THE DENSITY ESTIMATION PROBLEM. Given a function $f:[a,b] \to R$ satisfying the conditions

$$f(t) \geq 0 \quad \forall t \in [a,b]$$

and

$$\int_a^b f(t)dt = 1$$

define $F:[a,b] \to R$ by

$$F(s) = \int_a^s f(t)dt.$$

Suppose that the random variable y is uniformly distributed on the interval $[0,1]$, then the random variable

$$x = F^{-1}(y)$$

is distributed according to the probability density function f. In this way each random variable is characterized by its probability density function.

By the density estimation problem we mean given only the random samples $x_1,\ldots,x_n$ estimate, f, the unknown probability density function which gave rise to this sample. A detailed introduction to this problem and the various approaches suggested for its solution can be found in Tapia and Thompson (1978). In the following sections we present two successful approaches for the one dimensional problem and then consider various strategies for extending these approaches into efficient multidimensional estimation procedures.

[1] This work was sponsored by the U.S. Army Research Office under Grant DAAG29-78-G-0187

2. KERNEL DENSITY ESTIMATION. Besides the standard histogram estimator, perhaps the most common nonparametric solution to the density estimation problem is the kernel estimation procedure first considered by Rosenblatt (1956) and extensively developed and explicated by Parzen (1962). These estimators are of the form

$$\hat{f}_n(x) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h_n} K\left(\frac{x-x_i}{h_n}\right) \tag{1}$$

where the kernel K is a fixed probability density function (i.e., K is a nonnegative member of L^1 which integrates to one) and $\{h_n\}$ is a sequence of constants which do not depend on the sample points. The constant h_n is often referred to as the window width or scaling parameter or smoothing parameter and essentially controls the support of $K\left(\frac{x-x_i}{h_n}\right)$. Rosenblatt (1956) constructed the "shifted histogram" which can be shown to be of the form (1) with the specific kernel

$$K(x) = \begin{cases} 1/2 & |x| \leq 1 \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Kernel density estimation is an incredibly simple and straightforward tool. In one dimension its main drawback is the fact that the final product is quite sensitive to the choice of the smoothing parameter h_n. Specifically if h_n is too large the estimate $\hat{f}_n$ tends to be oversmoothed; while if h_n is too small the estimate tends to be excessively wiggly. A good example of this phenomenon is given in pages 60-66 of Tapia and Thompson (1978). Kernel estimators are not, in general, robust against poor choices of h_n. Accordingly, an interactive approach is usually pursued. The user chooses a large value of h_n and then decreases it until the estimate developes an excessive amount of wiggles. The optimal choice is taken to be the smallest value of h_n which did not produce excessive wiggles. While kernel density estimation is a simple tool it is only as effective as the procedure one employs for choosing the smoothing parameter h_n. For this reason there has been considerable activity in the construction of procedures which lead the user to a satisfactory choice for h_n. Along these lines we mention Scott, Tapia and Thompson (1977), Wahba (1978), Silverman (1978) and the recent study of several of these procedures performed by Factor and Scott (1979).

A second active research activity in the area of kernel density estimation concerns the convergence rate of these estimators. Let f_n be an estimate of the probability density function f which gave rise to the random sample $x_1,\ldots,x_n$. By the mean squared error of $f_n(x)$ we mean

$$\text{MSE}(f_n(x)) = E[(f_n(x) - f(x))^2] \tag{3}$$

where E denotes expectation. Furthermore, we say that f_n is a consistent estimator if $\text{MSE}(f_n(x)) \to 0$ as $n \to \infty$ for all x in the domain of f.

Parzen (1962) established that under mild conditions including the conditions

(4) $$h_n \to 0 \text{ and } nh_n \to \infty \text{ as } n \to \infty$$

the kernel estimator (1) is consistent and

(5) $$MSE(\hat{f}_n(x)) = O(n^{-4/5}) .$$

The Cramér-Rao bound says essentially that any nonparametric estimator will have the property that its MSE as a function of n cannot go to zero faster than n^{-1}, i.e., a nonparametric estimator f_n with the property that

(6) $$MSE(f_n(x)) = O(n^{-1})$$

would be optimal. It is interesting to observe that the standard histogram estimator f_n^h can be implemented in a manner which leads to

(7) $$MSE(f_n^h(x)) = O(n^{-2/3}) .$$

For details see Chapter 2 of Tapia and Thompson (1978). Hence from the standpoint of convergence rate the kernel estimators (including of course Rosenblatt's shifted histogram) are superior to the histogram estimator; however, they are not optimal in the sense of the Cramér-Rao bound.

Recently there has been considerable activity in trying to modify or "improve" the kernel estimator so as to obtain a convergence rate better than $O(n^{-4/5})$. Toward this end Davis (1975) considered working with the sinc function as the kernel, i.e.,

(8) $$K(x) = \sin x/\pi x .$$

Observe that in this case $K(x) \notin L^1(-\infty,\infty)$ and $K(x)$ is not nonnegative; although it does integrate to one. The kernel density estimation procedure with this kernel gives a MSE of order $\log(n)/n$ which is close to the optimal order of n^{-1}. The key question here is how badly do the negative lobes in this kernel influence the estimate. It would be hoped that for samples of moderate size (e.g. 200-400) there would be little deterioration in the quality of the estimate; however, this is not always the case (see pages 78-82 of Tapia and Thompson (1978)).

Very recently Scott and Terrell (1979) have observed that if instead of dropping the nonnegativity constraint one drops the constraint that the estimate integrate to one, then it is possible to obtain consistent kernel density estimators which have convergence rates arbitrarily close to the optimal rate of $O(n^{-1})$.

The question has arisen, e.g. Wagner (1975) and Breiman, Meisel and Purcell (1977) as to whether one might profit from a more general choice of smoothing parameters $\{h_n\}$. In particular it seems natural to replace (1) with

(9)
$$\hat{f}_n(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h_{ni}} K\left(\frac{x-x_i}{h_{ni}}\right)$$

where h_{ni} could depend on many things including the data point x_i and its neighbors. In this manner one would expect to get estimators which are very responsive to the data. However Tapia, Thompson and Trosset (1978) were able to demonstrate that for essentially any reasonable choice for the scaling parameters h_{ni}, the optimal convergence rate that can be obtained is still of the order of $n^{-4/5}$. This means that in terms of convergence rate the more general strategies for choosing the smoothing parameters offer nothing.

In summary we see that kernel density estimation (1) is simple and straightforward but sensitive to the choice of the smoothing parameters. It is not optimal in terms of convergence rate as dictated by the Cramér-Rao bound. Moreover its convergence rate cannot be improved without dropping either the nonnegativity constraint or the constraint that the estimate integrate to one.

3. MAXIMUM PENALIZED LIKELIHOOD DENSITY ESTIMATION. Consider the restricted Sobolev space

(10)
$$H_0^s(a,b) = \{f : f^{(j)} \in L^2(a,b),\ j = 0,\ldots,s \text{ and } f^{(j)}(a) = f^{(j)}(b) = 0,\ j = 0,\ldots,s-1\},$$

with inner product

(11)
$$\langle f,g\rangle_{H_0^s} = \int_a^b f^{(s)}(t)g^{(s)}(t)dt .$$

Given the random sample $x_1,\ldots,x_n$ by the maximum penalized likelihood estimate (MPLE) of the unknown probability density function which gave rise to this sample we mean any solution of the following infinite dimensional constrained optimization problem

(12) $$\text{maximize } L(V) = \prod_{i=1}^{n} V(x_i)\exp(-\alpha_n \langle V,V\rangle_{H_0^s}); \ (\alpha_n > 0)$$

subject to

$$V \in H_0^s(a,b), \int_a^b V(t)dt = 1 \text{ and } V(t) \geq 0 \ \forall t \in [a,b].$$

The suggestion of penalizing the likelihood is due to Good (1971) and Good and Gaskins (1971). The particular formulation given by (12) and the proof that problem (12) has a unique solution which is a polynomial spline is due to de Montricher, Tapia and Thompson (1975). All this material in detailed form can be found in Chapters 3,4 and 5 of Tapia and Thompson (1978). The constant α_n is analogous to the smoothing parameter h_n in the case of kernel density estimation.

In order to solve problem (12) we choose $s = 1$ and introduce a finite dimensional approximation by restricting our attention to S the finite dimensional subspace of $H_0^1(a,b)$ given by the linear splines with knots at this mesh points $a = t_0 < t_1 < \ldots < t_m < t_{m+1} = b$. We may as well choose an equally spaced mesh and let $h = t_{i+1} - t_i$. A typical member of S would look like

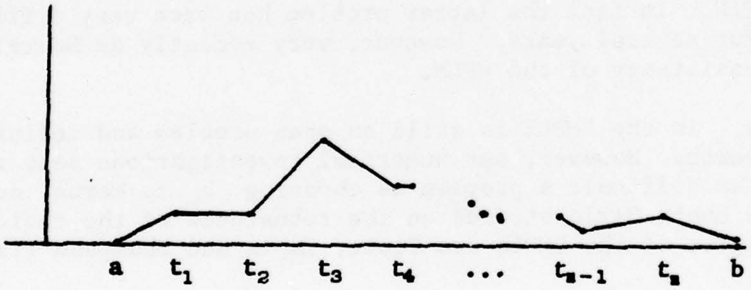


Figure 1

Let u denote our linear spline and $u_i = u(t_i)$. Then

(13) $$\langle u,u\rangle_{H_0^1} = \int_a^b u'(t)^2 dt = h^{-1} \sum_{i=0}^{m} (u_{i+1} - u_i)^2 ,$$

(14) $$\int_a^b u(t)dt = h \sum_{i=1}^{m-1} u_i \text{ , and}$$

(15) $$u(t) \geq 0 \Leftrightarrow u_i \geq 0, \ i = 1,\ldots,m .$$

With these observations we see that problem (12) restricted to S becomes

$$\text{(16)} \qquad \text{maximize} \prod_{i=1}^{n} u(x_i)\exp(-\alpha_n h^{-1}\sum_{i=0}^{m}(u_i - u_{i-1})^2)$$

$$\text{subject to} \quad h\sum_{i=1}^{m} u_i = 1$$

$$\text{and} \qquad u_i \geq 0,\ i = 1,\dots,m\ .$$

Notice that we are requiring $u_0 = u_{m+1} = 0$. The linear spline solution of problem (16) is referred to as the discrete maximum penalized likelihood estimator or DMPLE.

The development of this numerical implementation which gives the DMPLE as an approximation to the MPLE is due to Scott, Tapia and Thompson (1978a) and is also described in Chapter 5 of Tapia and Thompson (1978). They demonstrated that the DMPLE was consistent and for a fixed sample it converged to the MPLE for this sample as the mesh spacing h went to zero. This does not imply consistency of the MPLE. In fact the latter problem has been very difficult and evaded solution for several years. However, very recently de Montricher (1979) established consistency of the MPLE.

The choice of α_n in the DMPLE is still an open problem and definitely requires further research. However, our numerical investigations seem to imply that it is not as difficult a problem as choosing h_n in kernel density estimation. For some Monte Carlo studies on the robustness of the choice of α_n and also some examples of the DMPLE see Scott, Tapia and Thompson (1978a).

In summary we see that the main disadvantage of DMPLE is that it requires the solution of an optimization problem. By working with the negative log of the penalized likelihood functional we see that our optimization problem is a convex programming problem with dimension equal to the number of mesh points and fortunately not the number of sample points. For this seemingly large amount of work we do obtain reasonable estimates which are robust with respect to the choice of any parameters inherent in the formulation. Moreover, we do have the flexibility of immediately incorporating additional information into the model in the form of additional constraints.

4. DENSITY ESTIMATION IN HIGHER DIMENSIONS. There is a very serious need for effective and efficient density estimation procedures in higher dimensions in many statistical applications including the areas of

(i) pattern recognition
(ii) classification
(iii) simulation
(iv) modeling
(v) picture drawing.

Moreover it has been our experience that a straightforward approach using either kernel density estimation or discrete maximum penalized likelihood density estimation (DMPLE) will fail from the point of view of reasonable amount of computer time at dimensions greater than 3 or 4. We should emphasize that from a theoretical point of view it should be clear that it is a straightforward matter to formulate the kernel density estimation and DMPLE procedures in higher dimensions. Our point is that this straightforward formulation will simply lead to an algorithm which would be too costly in dimensions above 3 or 4.

Below we outline several procedures that merit further investigation and can be used in conjunction with kernel density estimation and/or DMPLE to effectively solve higher dimensional problems.

Reduction of the Dimension of the Data.

Often times we see the situation where the random samples bunch or collect on a manifold of lower dimension. What would be needed here is a procedure for both identifying and parameterizing this lower dimensional manifold. For example the data displayed in Figure 2 below is clearly two dimensional; however, it is very close to representing a one-dimensional manifold.

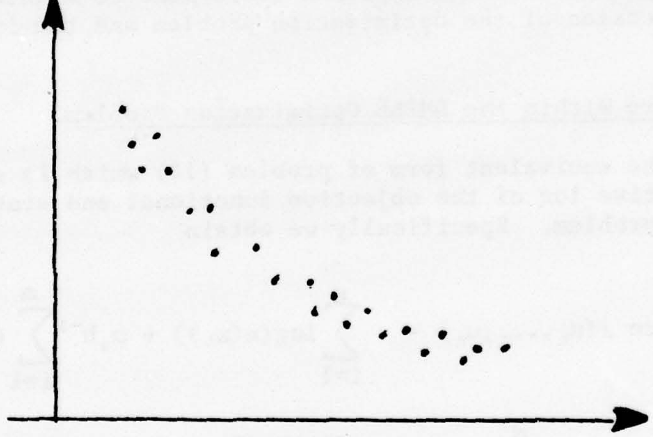


Figure 2

This general problem is not new. Moreover, its general solution is probably extremely difficult and maybe even impossible. However, satisfactory progress could be expected in specific applications.

Pseudo-independence Data Transformations.

It is often very efficient to approximate a multidimensional problem by a sequence of one-dimensional problems. When it is possible the amount of work grows linearly with the dimension of the problem and higher dimensional problems are not out of reach. In the present application we consider the situation when a probability density function of many variables can be approximated by a product of probability density functions of a single variable. One such approach is the pseudo-independence algorithm introduced by Bennett, de Figueiredo and Thompson (1974) and applied to multidimensional multimodel data in Scott, Tapia and Thompson (1978b). A brief description is also given in Chapter 5 of Tapia and Thompson (1978). Essentially in this approach the eigenvectors of the estimated covariance matrix are used to provide a pseudo-independent linear transformation. Then an algorithm for estimating one-dimensional densities is applied to each of the pseudo-independent dimensions. The underlying motivation here is the fact that the multidimensional Gaussian probability density function is the product of the one-dimensional marginal densities when the covariance matrix is diagonal. We feel that the procedure is not fully understood and offers some promise. For somewhat more detail and an interesting application and comparison with a straightforward approach see Scott, Tapia and Thompson (1978b). We leave this approach with the satisfying comment that the dimension of the transformation required is not the number of data points but the dimension of the data (which usually is small e.g. less than 20).

Selective Mesh Spacing Within DMPLE.

Since the dimension of the optimization problem that is solved in obtaining the DMPLE is equal to the number of mesh points it seems reasonable to attempt to construct a method for selecting the mesh points as a function of the data. In this way we would not produce many mesh points in areas where there is little or no data. Hopefully such an approach would lead to a balance or equilibrium between the dimension of the optimization problem and the information contained in the data.

Special Structure Within the DMPLE Optimization Problem.

Consider the equivalent form of problem (16) which is obtained by taking the negative log of the objective functional and stating the problem as a minimization problem. Specifically we obtain

$$(17) \qquad \text{minimize } J(u_1,\ldots,u_m) = -\sum_{i=1}^{n}\log(u(x_i)) + \alpha_n h^{-1}\sum_{i=1}^{m}(u_i - u_{i-1})^2$$

$$\text{subject to } \; h\sum_{i=1}^{m} u_i = 1$$

$$\text{and} \qquad u_i \geq 0, \quad i = 1,\ldots,m$$

Problem (17) is a convex programming problem which is rich in special structure. Specifically it has simple nonnegativity bounds on the variables and one linear constraint where the coefficients are all the same and positive. Moreover the Hessian matrix of J is positive definite and tridiagonal. In fact, it is actually a Minkowski matrix. Namely the off-diagonal elements are negative and the matrix is symmetric and positive definite. It seems then that a general purpose algorithm applied to this optimization problem would be extremely inefficient. There have been recent advances in the optimization literature for problems with similar structure. It follows that an optimization algorithm could be developed which would be very efficient and exploited this rich structure.

References

Bennett, J.O., de Figueiredo, R.J.P., and Thompson, J.R. (1974). "Classification by means of B-spline potential functions with applications to remote sensing," The Proceedings of the Sixth Southwestern Symposium on Systems Theory, FA3.

Breiman, L., Meisel, W., And Purcell, E. (1977). "Variable kernel estimates of multivariate densities," Technometrics, Vol. 19, 135-144.

Davis, K.B. (1975). "mean square error properties of density estimates," Annals of Statistics, Vol. 3, 1025-1030.

de Montricher, G. (1979). "On the consistency of the maximum penalized likelihood estimator," in preparation.

de Montricher, G., Tapia, R.A., and Thompson, J.R. (1975). "Nonparametric maximum likelihood estimation of probability densities by penalty function methods," Annals of Statistics, Vol. 3, 1329-1348.

Factor, L., and Scott, D. (1979). "A Monte Carlo study of 3 data based nonparametric probability density estimators," submitted.

Good, I.J. (1971). "A nonparametric roughness penalty for probability densities." Nature, Vol. 229, 29-30.

Good, I.J., and Gaskins, R.A. (1971). "Nonparametric roughness penalties for probability densities." Biometrika, Vol. 36, 149-176.

Rosenblatt, M. (1956). "Remarks on some nonparametric estimates of a density function." Annals of Mathematical Statistics, Vol. 27, 832-839.

Parzen, E. (1962). "On estimation of a probability density function and mode," Annals of Mathematical Statistics, Vol. 33, 1065-1076.

Scott, D.W., Tapia, R.A., and Thompson, J.R. (1977). "Kernel density estimation revisited," Nonlinear Analysis, Vol. 1, 339-372.

Scott, D.W., Tapia, R.A., and Thompson, J.R. (1978a). "Nonparametric probability density estimation by discrete maximum penalized-likelihood criteria," to appear Annals of Statistics.

Scott, D.W., Tapia, R.A., and Thompson, J.R. (1978b). "Multivariate density estimation by discrete maximum penalized likelihood methods," Graphical Representation of Multivariate Data, Ed. P. Wang, Academic Press, New York.

Scott, D.W., and Terrell, G. (1979). "On improving the rate of convergence of nonnegative kernel density estimators," submitted.

Silverman, B.W. (1978). "Choosing the window width when estimating a density," Biometrika, Vol. 65, 1-11.

Tapia, R.A., and Thompson, J.R. (1978). Nonparametric Probability Density Estimation. Johns Hopkins University Press, Baltimore, Maryland.

Tapia, R.A., Thompson, J.R., and Trosset, M.W. (1978). "On the optimal scaling of kernel density estimators," submitted.

Wagner, T.J. (1975). "Nonparametric estimates of probability densities." IEEE Trans. Information Theory. IT-21, 438-440.

Wahba, G. (1978). "Data based optimal smoothing of orthogonal series density estimates," University of Wisconsin, Statistics Dept. TR #509.

IMPLEMENTATION OF THE GENERALIZED REDUCED GRADIENT METHOD FOR NONLINEAR PROGRAMMING PROBLEMS

Gary A. Gabriele*

Army Institute for Research in Management
Information and Computer Science
Atlanta, Georgia

ABSTRACT. This paper describes an algorithm for the solution of nonlinear programming problems based on the generalized reduced gradient method of Abadie. Attention will first be focused on the theoretical aspects of the method and secondly, on its implementation. Computational experience with the method has shown it to be robust and reliable for a large class of problems. Results on three different engineering applications will be presented.

1. INTRODUCTION. We will concern ourselves with the class of problems known as the nonlinear programming (NLP) problem. The general NLP problem can be stated in the following form:

$$\text{Minimize} \quad f(x) \qquad x=[x_1,x_2,\ldots,x_N] \tag{1}$$

$$\text{subject to} \quad g_k(x) \geq 0 \qquad k=1,2,\ldots,K \tag{2}$$

$$h_\ell(x) = 0 \qquad \ell=1,2,\ldots,L \tag{3}$$

where

x = a column vector of design variables
$f(x)$ = objective function
$g_k(x)$ = inequality constraints; these functions delimit regions in the design space
$h_\ell(x)$ = equality constraints; these functions require specific combinations of the design variables.

In the NLP problem it is assumed that the functions f, g, and h are nonlinear, and for the purposes of our discussion we will assume them to be differentiable. Although many strategies have been suggested to solve the NLP problem, none has proved to be successful in all cases. The Generalized Reduced Gradient method, however, has shown {1,2} itself to be the most efficient and reliable method yet available.

The author presented this paper at the 1978 Numerical Analysis and Computers Conference.

2. GENERALIZED REDUCED GRADIENT - THEORY. The reduced gradient method was originally presented by Wolfe {3,4} for problems possessing a nonlinear objective function and linear constraints. A generalization of Wolfe's method to accommodate nonlinearities in the constraints was accomplished by Abadie and Carpentier {5}.

For purposes of the generalized reduced gradient method the NLP problem of (1)-(3) can be restated in the following form:

$$\text{Minimize} \quad f(x) \qquad x=[x_1,x_2,\ldots,x_N]^T \tag{4}$$

$$\text{subject to} \quad h_m(x) = 0 \qquad m=1,2,\ldots,M \tag{5}$$

$$A \leq x \leq B \tag{6}$$

The N x 1 vectors A and B represent upper and lower bounds on the design vector x. The inequality constraints (2) have been included as equality constraints through the use of slack variables. The slack variables are folded into the problem as non-negative design variables, and the parameter M now represents the total number of constraints,

$$M = L + K \tag{7}$$

It should be stressed that the constraints of (5) represent functional constraints, with variable bounds included separately in (6). The distinction is made so that variable bounds may be treated separately.

Consider the following strategy. Divide the design variables x into two classes which we shall label as decision and state variables,

$$x = [z,y]^T \tag{8}$$

$$z = [z_1,z_2,\ldots,z_Q]^T \text{ ; decision variables} \tag{9}$$

$$y = [y_1,y_2,\ldots,y_M]^T \text{ ; state variables} \tag{10}$$

$$Q = N-M \tag{11}$$

We shall choose the decision variables to be completely independent and the state variables shall become slaves to the decision variables used to satisfy the constraints.

The following notation shall be useful in the discussion to follow:

$$f_z = [\frac{\partial f}{\partial z_1}, \frac{\partial f}{\partial z_2}, \ldots, \frac{\partial f}{\partial z_Q}]^T$$

$$f_y = [\frac{\partial f}{\partial y_1}, \frac{\partial f}{\partial y_2}, \ldots, \frac{\partial f}{\partial y_M}]^T$$

$$\frac{\partial h}{\partial z} = \begin{bmatrix} \frac{\partial h_1}{\partial z_1} & \frac{\partial h_1}{\partial z_2} & \cdots & \frac{\partial h_1}{\partial z_Q} \\ \cdot & & & \\ \cdot & & & \\ \cdot\,\cdot & & & \\ \frac{\partial h_M}{\partial z_1} & \frac{\partial h_M}{\partial z_2} & \cdots & \frac{\partial h_M}{\partial z_Q} \end{bmatrix} \qquad \frac{\partial h}{\partial y} = \begin{bmatrix} \frac{\partial h_1}{\partial y_1} & \frac{\partial h_1}{\partial y_2} & \cdots & \frac{\partial h_1}{\partial y_M} \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ \frac{\partial h_M}{\partial y_1} & \frac{\partial h_M}{\partial y_2} & \cdots & \frac{\partial h_M}{\partial y_M} \end{bmatrix}.$$

Introducing the decision and state variable distinction into the first variation of f(x) and h(x) we obtain

$$df = f_z^T\, dz + f_y^T\, dy \tag{12}$$

$$dh = \frac{\partial h}{\partial z}\, dz + \frac{\partial h}{\partial y}\, dy = 0 \tag{13}$$

Solving (13) for dy yields,

$$dy = \frac{\partial h}{\partial y}^{-1} \frac{\partial h}{\partial z} dz(-1) \tag{14}$$

Substituting (14) into (12) and rearranging will yield the following linear approximation to the reduced gradient,

$$\frac{df}{dz} = f_z^T - f_y^T\, \frac{\partial h}{\partial y}^{-1}\, \frac{\partial h}{\partial z} \tag{15}$$

The reduced gradient defines the rate of change of the objective function with respect to the decision variables with the state variables adjusted to maintain feasibility. Equation (15) gives the changes necessary in the states for a given change in the decisions. Geometrically the reduced gradient can be described as a projection of the original N- dimensional gradient onto the Q- dimensional feasible region described by the decision variables.

A necessary condition for the existence of a minimum of an unconstrained nonlinear function requires the elements of the gradient vanish. Similarly, a minimum of the constrained nonlinear function occurs when the appropriate elements of the reduced gradient vanish. This conclusion can be verified by a comparison with the Kuhn-Tucker conditions {6} for the existence of a constrained relative minimum.

By first transforming the variable bounds into inequality constraints,

$$g_i(x) = x_i - a_i \geq 0 \tag{16}$$

$$g_{N+i}(x) = b_i - x_i \geq 0$$

we can form the following Lagrangian function,

$$L(x,u,w) = f(x) + \sum_{m=1}^{M} w_m h_m(x) - \sum_{j=1}^{2N} u_j g_j(x) \tag{17}$$

The following Kuhn-Tucker necessary conditions hold for a point to be a relative minimum x*,

$$\frac{\partial f^T}{\partial x} + \sum_{m=1}^{M} w_m^* \frac{\partial h_m}{\partial x} - \sum_{j=1}^{J} u_j^* \frac{\partial g_j}{\partial x} = 0 \tag{18}$$

and

$$h_m(x^*) = 0 \qquad m = 1,2,\ldots,M \tag{19}$$

$$g_j(x^*) \geq 0 \qquad j = 1,2,\ldots,J = 2N \tag{20}$$

$$u_j^* g_j(x^*) = 0 \qquad j = 1,2,\ldots,j = 2N \tag{21}$$

$$u_j^* \geq 0 \qquad j = 1,2,\ldots,j = 2N \tag{22}$$

$$w_m^* \neq 0 \qquad m = 1,2,\ldots,M \tag{23}$$

Introducing the decision and state variable distinction into (18) and decomposing we can obtain:

$$f_z^T + w^{*T} \frac{\partial h}{\partial z} - u^{*T} \frac{\partial g}{\partial z} = 0 \tag{24}$$

$$f_y^T + w^{*T} \frac{\partial h}{\partial y} - u^{*T} \frac{\partial g}{\partial y} = 0 \tag{25}$$

For reasons to be examined in the next section, a state variable is not allowed to be equal or sufficiently close to either of its bounds. Therefore, those elements of u^* corresponding to the state variables will be zero, as well as those elements of $\partial g/\partial y$ corresponding to the decision variables. These two facts combine to eliminate the last term of (25). Solving (25) for w* and substituting into (24) will produce the following expression:

$$f_z^T - f_y^T \frac{\partial h}{\partial y}^{-1} \frac{\partial h}{\partial z} - u^{*T} \frac{\partial g}{\partial z} = 0 \tag{26}$$

Rearranging we obtain

$$u^{*T} \frac{\partial g}{\partial z} = f_z^T - f_y^T \frac{\partial h}{\partial y}^{-1} \frac{\partial h}{\partial z} \tag{27}$$

We recognize the right hand side to be the reduced gradient, df/dz. By examining the left hand side the following conditions for a candidate point x to be optimal can be formulated.

$$\frac{df}{dz_i} \begin{cases} >0 \text{ if } z_i = a_i \\ <0 \text{ if } z_i = b_i \\ =0 \text{ if } a_i \leq z_i \leq b_i \end{cases} \tag{28}$$

for $i = 1, 2, \ldots, Q$

3. GENERALIZED REDUCED GRADIENT-ALGORITHM. The generalized reduced gradient algorithm to be described here solves the general NLP problem stated in (1) - (3). The user inputs his problem using that formulation and the algorithm performs a transformation to the form given in (4) - (6). The solution of this new formulation is then obtained by solving a series of reduced problems of the form.

$$\text{Minimize} \quad f(z) \tag{29}$$

$$\text{subject to} \quad A \leq z \leq B \tag{30}$$

The generalized reduced gradient algorithm is composed of the following major steps:

1. Specify the decision and state variable sets.

2. Using (15) calculate the elements of the reduced gradient.

3. Using the conditions (28) determine if the current point is optimal. If these conditions are satisfied the algorithm halts, otherwise we continue on to Step 4.

4. Calculate a search direction for the decision variables based on the results of Step 2, and a search direction for the states based on (14).

5. Minimize $f(x)$ along the search directions obtained in Step 4 and return to Step 1.

In the following sections, implementation and computational considerations for each step above will be discussed.

1. Specify Decision and State Sets. In most cases the initial selection of decision and state variables is arbitrary. The following conditions should be maintained; however,

(a) State variables should be chosen to insure that the matrix $\partial h/\partial y$ is nonsingular.

(b) State variables are subject to arbitrary adjustment in order to maintain feasibility. Therefore, any design variable that is sufficiently close to either of its bounds should be included in the decision state. This condition was necessary to the optimality proof leading to (28).

(c) Since they are included linearly in the constraints, all slack variables should be included in the state set. An exception to this would be given by (b), implying that the corresponding constraint is active.

2. Calculation of the Reduced Gradient. The elements of the reduced gradient are given by (15) with all partial derivatives calculated numerically or given analytically by the user. In the present implementation, numerical derivatives have shown to be reliable, and the following computationally efficient transformation technique is used,

(a) Define a transformation matrix D given by

$$D = \frac{\partial h}{\partial y}^{-1} \frac{\partial h}{\partial z} \tag{31}$$

(b) Evaluate f(x) at the current point

$$F_1 = f(z,y) \tag{32}$$

(c) To calculate the ith element of df/dz, increment the ith element of the decision vector by some small amount ε.

$$z_i = z_i + \varepsilon \tag{33}$$

and increment all M state variables simultaneously by the following prescription

$$y_m = y_m - \varepsilon D_{mi} \qquad m = 1,2,\ldots,M \tag{34}$$

The ith element of the reduced gradient becomes

$$\frac{df}{dz_i} = \frac{f(z_1,z_2,\ldots,z_i,\ldots,z_Q) - F_1}{\varepsilon} \tag{35}$$

This method requires (N-L) + 1 objective function evaluations versus N+1 evaluations for forward differencing.

3. Convergence. Using the elements of df/dz calculated above, a projected reduced gradient is formed from the following set of conditions

$$df/dz_i = 0 \begin{cases} \text{if } z_i = b_i \text{ and } df/dz_i < 0 \\ \text{if } z_i = a_i \text{ and } df/dz_i > 0 \end{cases} \tag{36}$$

If the L2 norm of the projected reduced gradient is less than some specified criteria, a constrained relative minimum has been obtained. The algorithm may also halt if the change in design variable values is sufficiently small, indicating no significant move could be made that reduces f(x).

4. Determine Search Directions. Since df/dz defines the gradient of the reduced problem (29)-(30), it may be used by itself as a search direction for the decision variables or used in conjunction with any of the gradient based unconstrained searching techniques. The conjugate gradient method of Fletcher and Reeves {7} was chosen because of its reduced storage requirements. Restart of the method occurs every Q+1 iterations or whenever the decision variable set is respecified. Variable metric methods have also been used with the generalized reduced gradient algorithm with much success {8,9}.

A search direction for the state variables can be determined from (14) using the search direction of the decision variables determined above. However, this only describes to linear terms the required adjustment to the states. Further adjustment will be necessary to maintain feasibility when the constraints are nonlinear.

5. Determine Local Minimum. The normal course of events in locating a minimum along a line consists of two phases, the first phase being to locate an initial bracket within which the minimum is known to exist. The second phase consists of narrowing this bracket to some known tolerance. The generalized reduced gradient method uses this same two-phase procedure with modifications to accommodate the use of state and decision variables.

Using the search directions determined in the previous step and some starting point $\{z,y\}^0$ the line search is performed in the following manner.

1) Set $k=0$; $f^0=f(z^0,y^0)$; $\alpha^0=0$; $\Delta\alpha=1/||P||$; $k=k+1$

2) $\alpha^k=\alpha^{k-1}+\Delta\alpha$

$$z_i^k = \begin{cases} b_i & \text{if } z_i^0+\alpha^k P(z_i) \geq b_i \\ a_i & \text{if } z_i^0+\alpha^k P(z_i) \leq a_i \\ z_i^k+\alpha^k P(z_i) & \text{otherwise} \end{cases} \qquad (37)$$

$$i = 1,2,\ldots,Q$$

and

$$y_m^k=y_m^0+\alpha^k P(y_m) \qquad (38)$$

$$m = 1,2,\ldots,M$$

where P(z) and P(y) represent the search directions of the decision and state variables respectively.

Because of nonlinearities arising in the constraint functions, the point $\{z,y\}^k$ is likely to be infeasible. Holding the decision variables constant, the state variables are adjusted to obtain a feasible point.

This is equivalent to solving M nonlinear equations ($h(x)=0$) in M unknowns (y). To accomplish this a modified Newton's method was used,

$$y^{t+1} == y^t - \frac{\partial h^{-1}}{\partial y} h(z^k, y^t) \qquad (39)$$

where

$\frac{\partial h^{-1}}{\partial y}$ = the initial inverse used to calculate the reduced gradient

$t = t^{th}$ iteration of Newton's method

Convergence occurs when all constraints are within some small neighborhood of zero. If convergence cannot be obtained quickly enough and $k=1$, then $\Delta\alpha$ is reduced and this step is repeated until $\Delta\alpha$ becomes too small, otherwise, the best point obtained thus far is excepted as the local minimum.

3) Evaluate the current values of the state variables, y^k, to determine if any lie outside their bounds. If no bounds are violated, we continue to the next step. Otherwise, a linear interpolation is performed between $\{z,y\}^{k-1}$ and $\{z,y\}^k$ to reduce α^k to the value where the nearest bound becomes active. Supplementary tests are performed to determine if the minimum lies at α^k or before it. If the minimum lies at α^k, then this point is accepted as the local minimum. If not, then a bracket on the minimum has been established and we move to step 5.

```
4) Set $f^k=f(z^k,y^k)$.
   For $k=1$;  $f^k<f^0$, set $k=k+1$ and go to step 2
               $f^k>f^0$, reduce $\Delta\alpha$ and go to step 2

   For $k=2$;  $f^k>f^{k-1}$ minimum bracketed, go to step 5
               $f^k<f^{k-1}$ update $f^{k-1}=f^k$, $\{z,y\}^{k-1}=\{z,y\}^k$
                      set $\Delta\alpha=2\Delta\alpha$, go to step 2
```

5) The interval $\{\alpha^o,\alpha^k\}$ brackets the local minimum. Reduction of this bracket can now be performed to determine to some specified satisfaction the location of the local minimum. Determination of each trial point should be accomplished using the strategy of step 2.

6 Respecify State and Decision Variables. We see in the previous step that it was possible to end a line search with a state variable equal to one of its specified bounds. To start a new interation the conditions set forth for state variables must be satisified. Those state variables that are bounded must be exchanged with decision variables that unbounded and will not cause $\partial h/\partial y$ to become singular.

Assume the y_s is a bound state variable. Abadie {5} suggests the following expression be maximized to determine the appropriate decision variable to exchange with y_s

$$\text{MIN } \{|D_{si}|(b_i - z_i) \text{ , } |D_{si}|(z_i - a_i)\} \qquad i = 1,2,\ldots,Q \qquad (40)$$

where

$$D_{si} = \text{element of } \frac{\partial h}{\partial y}^{-1} \frac{\partial h}{\partial z}$$

Another technique for the selection of new state variables is given by Lasdon, etal {9}. Their approach centers around avoiding ill-conditioning the $\partial h/\partial y$ matrix. No information has been published that points to the advantages that one approach might have over the other.

4. NUMERICAL EXPERIMENTS USING THE GENERALIZED REDUCED GRADIENT. Past studies {1,2} have shown the method described in this paper to be robust and reliable on a large class of problems. To demonstrate this claim, solutions to three differing engineering problems will be given. The first is a small scale design problem which will be referred to as the Welded Beam Problem. The second is an example of a medium-sized non-linear simulation problem of a synthetic natural gas plant and will be referred to as the SNG problem. The last problem is an example of structural optimization using the generalized reduced gradient and will be identified as the Structures Problem. Solutions to the first two problems were obtained on a CDC-6500 system with the last being done on a PDP 11/70. All partials required were obtained numerically on both machines.

Welded Beam Problem. This problem was originally proposed by Harold Keith {10} and later solved by Ragsdell and Phillips {11}. The structure is shown in Fig. 1. The objective is to find a feasible combination of h, ℓ, t, and b, such that the total cost is minimum. The values F and L are fixed and variable bounds exist on h, ℓ, t, and b. Five functional inequality constraints are imposed which limit the weld stress, bar bending stress, bar buckling load, bar deflection, and weld thickness. From a starting point of h=1, ℓ=7, t=4, and b=2, with cost \$15.82, the final solution of h=0.2455, ℓ=6.1960, t= 8.2730 and b=0.2455 with cost \$2.38 was obtained in five iterations. A total of 56 objective function evaluations and 191 constraint set evaluations was needed with approximately .5 sec of CPU time.

SNG Problem. The SNG problem was originally proposed by Spencer Shuldt {12} and is further documented in {2}. The problem possesses 48 design variables, one inequality and two equality constraints. Variable bounds exist for all design variables. The objective function is a highly nonlinear simulation of the cost incurred in running a fictitious synthetic natural gas plant. The constraints describe conditions that must exist to maintain operation and are of a similar functional nature as the objective function. This problem is representative of a medium to large nonlinear problem that leaves some variables unconstrained. Hence, performance can depend on the type of search direction employed. The generalized reduced gradient method using a conjugate gradient method required 161 iterations to reduce the objective function from 1862.30 to 863.39. The number of function evaluations required was 10100 with 13837 constraint set evaluations. CPU time was approximately one minute.

Structures Problem. A general mathematical programming formulation for spatial structures was originally proposed by Fox and Schmit {13}. The problem posed is to minimize the weight of structure subjected to specified loading conditions and constraints on member stress, strain and buckling. The design variables consist of mean diameter and thickness of each member (assuming annular cross section), stress in each member for each loading condition, and joint displacement. Using the Ramberg-Osgood stress-strain relationships produces a problem that possesses a relatively simple objective function with a large number of highly nonlinear constraints, both equality and inequality. In our example, the structure of fig. 2 resulted in 22 design variables, 14 equality constraints and 24 inequality constraints. Using aluminum, a starting weight of 821.34 lb was reduced to 125 lb in 15 iterations. The total number of objective function and constraint set evaluations was 213 and 1272 respectively. CPU time on the 11/70 was approximately five minutes, 30 seconds.

Additional numerical results with the program used to obtain these results are reported in Sandgren {2}.

The excellent performance demonstrated on these problems is a result of the ease and efficiency with which the generalized reduced gradient algorithm identifies and follows active constraints. The state and decision variable dichotomy helps to reduce the constrained nonlinear programming problem to a simpler, more readily solved problem. By using the dichotomy the problem becomes, in simple terms, a search in the reduced unconstrained feasible region of the decision variables while maintaining feasibility through state variable adjustment.

5. REFERENCES:

1. Colville, A. R., "A Comparative Study of Nonlinear Programming Codes," Proceedings of the Princeton Symposium on Mathematical Programming, Kuhn, H. W., ed., Princeton, N. J., 1970, pp 487-501.

2. Sandgren, E., "The Utility of Nonlinear Programming Algorithms, Parts 1 and 2," The Modern Design Series, Mechanical Engineering Design Group, Purdue University, 1977.

3. Wolfe, P., "Methods for Linear Constraints," Nonlinear Programming, Abadie, J., ed., North Holland, Amsterdam, 1967, pp. 99-131.

4. Wolfe, P., "Methods of Nonlinear Programming," Recent Advances in Mathematical Programming, Graves, R. L., and Wolfe, P., eds., McGraw-Hill, New York, 1963, pp. 76-77.

5. Abadie, J., and Carpentier, J., "Generalization of the Wolfe Reduced Gradient Method to the Case of Nonlinear Constraints," Optimization, Fletcher, R., ed., Academic Press, 1969, p. 37.

6. Kuhn, W. W., and Tucker, A. W., "Nonlinear Programming," Proceedings, 2nd Berkeley Symposium on Mathematical Statistics and Probability, University of California Press, Berkeley, 1951, pp. 481-492.

7. Fletcher, R., and Reeves, C. M., "Function Minimization by Conjugate Gradients," Computer Journal, Vol. 6, No. 2, 1963, pp. 163-168.

8. LaFrance, L. J., "The Development of a Nonlinear Constrained Optimization Algorithm for Application to Simulation Systems," Ph.D Dissertation, Mechanical Engineering Dept., Purdue University, August, 1975.

9. Lasdon, L. S., et al., "Design and Testing of a Generalized Reduced Gradient Code for Nonlinear Programming," ACM Trans. on Mathematical Software, Vol. 4, No. 1, March 1978, pp. 34-50.

10. Keith, Harold D., "Optimization Techniques in Design," ASME, No. 69-DE-14, 1969.

11. Ragsdell, K. M., and Phillips, D. T., "Optimal Design of a Class of Welded Structures Using Geometric Programming," ASME, No. 75-DET-86.

12. Shuldt, S., Honeywell Corporate Research Center, Bloomington, Minn., personal communication.

13. Fox, R. L., and Schmit, L. A., "Advances in the Integrated Approach to Structural Synthesis," Journal of Spacecraft, Vol. 3, No. 6, June 1966, pp 858-866.

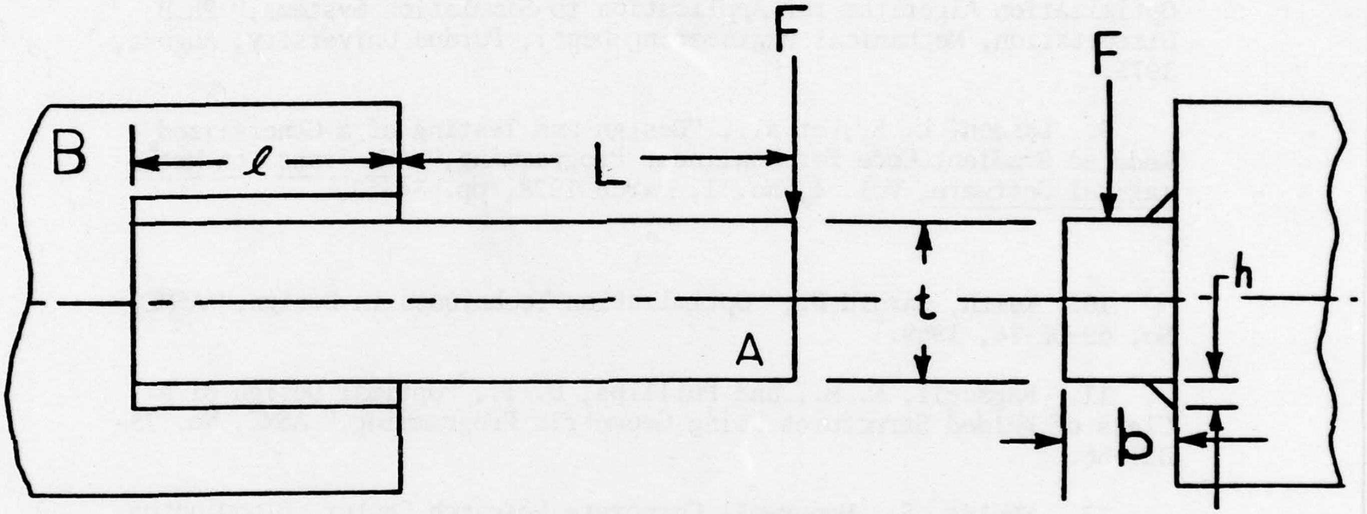
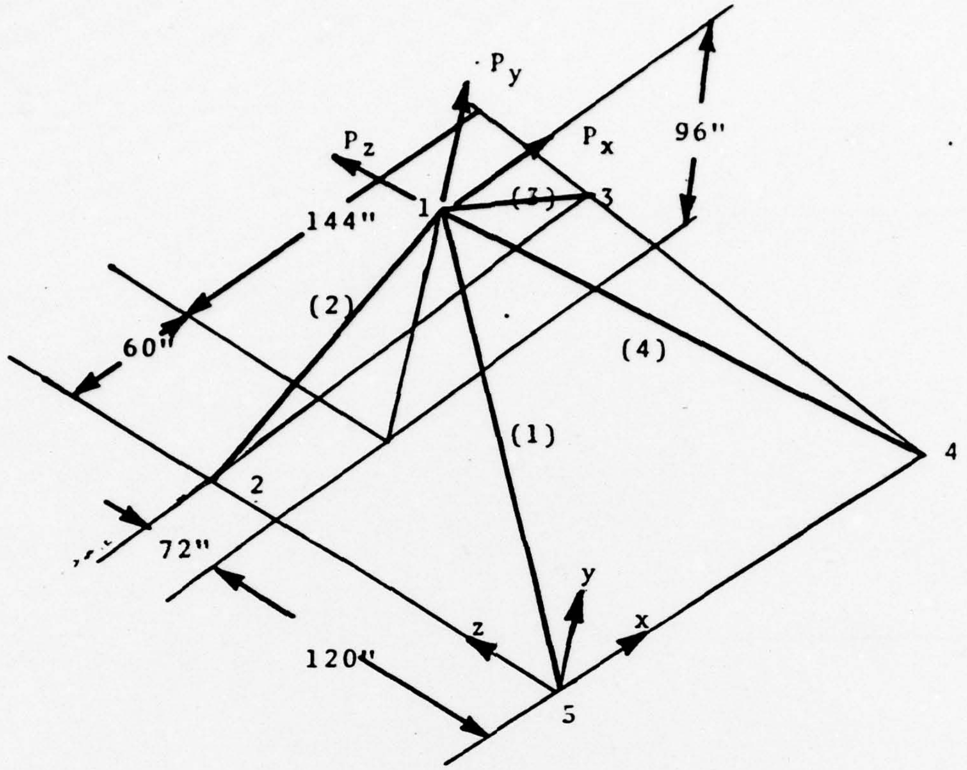


FIGURE 1. WELDED BEAM PROBLEM



MATERIAL: ALUMINUM, $E=10^7$ psi, ρ=.lpci
STRESS LIMITS: ±25000 psi on all members

LOADING CASE 1: SINGLE LOAD P_x=10K P_y=20k P_z=-60K

LOADING CASE 2: SINGLE LOAD P_x=40k P_y=]00k P_z=-30k

FIGURE 2. SIMPLE SPATIAL STRUCTURE PROBLEM

ROSTER OF REGISTRANTS

Agee, William S.	WSMR
Bash, Dave	USACACDA
Bates, Carl B.	USACAA
Bentley, Bedford T.	BRL
Boggs, Paul T.	ARO
Carling, Robert	Ottawa, Canada
Carrillo, Jesus E.	USA TRASANA
Castillo, Cesar	WSMR
Chandra, Jagdish	ARO
Chen, Peter C. T.	Watervliet Arsenal
Cheney, E. Ward	University of Texas at Austin
Chuvala, Ray	ARRADCOM, Dover, New Jersey
Dale, Richard	WSMR
Dalton, Oren N.	WSMR
de Boor, Carl	MRC
Doedel, Eusebius J.	Vanderbilt University, TN
Ferguson, Warren	MRC
Garfinkel, Gerald	WSMR
Glimm, James	Rockefellar University, NY
Golub, Gene H.	Stanford University, CA
Gomez, Jose E.	WSMR
Gose, J. B.	New Mexico State University
Goulet, Bernard N.	AMSAA, Aberdeen, Maryland
Graves, James A.	WSMR
Green, Robert E.	WSMR
Guard, Keith	New Mexico State University
Hausner, Arthur	HDL, Adelphi, Maryland
Hirschberg, Morton A.	BRL
Hopp, Theodore	HDL
Hoppe, George W.	National Guard Bureau
Klema, Virginia	Massachusetts Inst. of Technology
Kuehl, Gary G.	BRL
Kuykendall, Patrick	
Kuzanek, Jerry F.	WSMR
Li, T. Y.	MRC
McLaughlin, Dale	WSMR
McLeod, Robin	New Mexico State University
Nohel, John A.	MRC
Norman, James H.	Las Cruces, NM
Ohmstede, William D.	WSMR

Oliver, Carl E.	AFOSR
Ontiveros, Anita	USA TRASANA
Ott, Garland H.	NSWS, Dahlgren, Virginia
Phillips, Lindsay F.	USA TRASANA
Plemmons, Robert J.	The University of Tennessee
Rall, Louis B.	MRC
Reddien, G. W.	Vanderbilt University, TN
Roache, Patrick J.	Ecodynamics Research Assoc., Inc.
Sassenfeld, Helmut M.	WSMR
Schlenker, George	ARRCOM, Rock Island, IL
Shepherd, W. L.	WSMR
Smith, Philip W.	Texas A&M University
Starner, John	WSMR
Tapia, Richard A.	Rice University, Texas
Taylor, S. M.	BRL
Thomas, John D.	New Mexico State University
Thompson, James L.	TARADCOM
Thompson, James R.	Rice University, TX
Turner, Robert	WSMR
Wang, Chia Ping	USA NARADCOM
Wu, Julian J.	Watervliet Arsenal
Wolff, Stephen S.	BRL
Zacks, Shelemyahu	Case Western Reserve Univ., OH
Zoltani, Csaba K.	BRL

AD-A074 800   ARMY RESEARCH OFFICE RESEARCH TRIANGLE PARK NC   F/G 5/9
PROCEEDINGS OF THE 1979 ARMY NUMERICAL ANALYSIS AND COMPUTERS C--ETC(U)
SEP 79
UNCLASSIFIED   ARO-79-3   NL

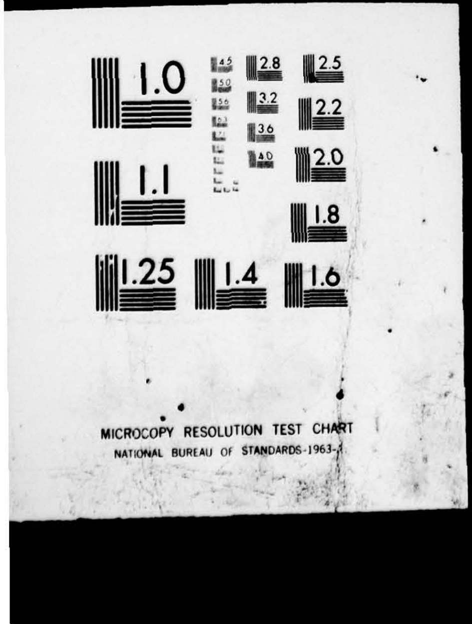
6 OF 6
AD
A074 800

END
DATE
FILMED
11-79
DDC

SIFIED

OF 6

74800



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

SECURITY CLA

1. REPORT NU

ARO Repo

4. TITLE (and S

Proceedi
and Comp

7. AUTHOR(s)

9. PERFORMIN

11. CONTROLLI

Army Mat
the Chie

14. MONITORING
Army Res
ATTN: D
P. O. Bo
Research

DISTRIBUTIO

Approved f
report are
unless so

17. DISTRIBUTIO

18. SUPPLEMENT

This is a
Computers
engineerin

19. KEY WORDS (

nonlinear
hit and ki
impact pre
surface mi
Ott functi
Repro-model
Variationa
Normal dis
Line of si
Robust Sof
Sparce matr
Numerical
Bacterial
Gun Motion

DD FORM 1 JAN 73 147

U S GOVERNMENT

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ARO Report 79-3	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Proceedings of the 1979 Army Numerical Analysis and Computers Conference		5. TYPE OF REPORT & PERIOD COVERED Interim Technical Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s)		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Army Mathematics Steering Committee on behalf of the Chief of Research, Development and Acquisition		12. REPORT DATE September 1979
		13. NUMBER OF PAGES 482
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Army Research Office ATTN: DRXRO-MA P. O. Box 12211 Research Triangle Park, NC 27709		15. SECURITY CLASS. (of this report)
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited. The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

This is a technical report resulting from the 1979 Army Numerical Analysis and Computers Conference. It contains papers on computer aided design and engineering as well as papers on numerical analysis.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

nonlinear least squares
hit and kill probabilities
impact predictions
surface miss distance program
Ott functions
Repro-modeling
Variational models
Normal distribution parameters
Line of sight data
Robust Software
Sparce matrices
Numerical evaluation of integrals
Bacterial survival equations
Gun Motion Analysis

fluids near a sliding corner
robust regression
Agee's smoothing method
GIFT computer code
Angle measurement equipment
Existence theorems
Fluid discontinuities
Optimal control
Nonuniform meshes
Approximations
Jacobi matrices
Curve fitting
Density estimation
Gradient methods

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

U S GOVERNMENT PRINTING OFFICE 1979-644-301